Toward an Efficient Generation of ISO 26262 Automotive Safety Analyses

Abraham Cherfi

Toward an Efficient Generation of ISO 26262 Automotive Safety Analyses

Vers une Génération Efficace d'Analyses de Sûreté de Fonctionnement dans le Cadre du Déploiement de l'ISO 26262

PRÉSENTÉE LE 2 Juillet 2015

A l'Ecole Polytechnique (Paris-Saclay)

Ecole Doctorale Polytechnique (EDX)

ÉCOLE POLYTECHNIQUE

POUR L'OBTENTION DU GRADE DE DOCTEUR DE L'ECOLE POLYTECHNIQUE

PAR

Abraham CHERFI

Acceptée sur proposition du jury :

Président de Jury :	Leila KLOUL	Laboratoire PRiSM, UVSQ
Rapporteurs :	Karama KANOUN	LAAS-CNRS, Toulouse
	Jean-Marc FAURE	LURPA, ENS Cachan
Directeur de thèse :	Antoine RAUZY	LGI, Centrale-SUPELEC
Examinateur :	Michel LEEMAN	GEEDS, Valeo

Remerciements

Au terme de ce travail, c’est avec émotion que je tiens à remercier tous ceux qui ont contribué à la réalisation de ce projet, de près ou de loin.

J'exprime mes profonds remerciements à mon directeur de thèse et mon encadrant industriel − le professeur Antoine Rauzy et Michel Leeman − pour l'aide compétente qu'ils m'ont apportée et leur patience. Leurs yeux critiques et leurs connaissances m’ont été très précieux pour structurer mes travaux durant ces trois années.

Ensuite, je tiens à remercier Stéphane pour m’avoir accueilli au sein de son équipe, et mes collègues pour leurs soutien et encouragements : Ludovic, Nieves, Gilles, Elmahdi, Nabila, Styven, Riad, Tatiana, Michel, Pierre-Antoine… sans qui ces trois années n’auraient jamais été aussi agréables.

Je remercie aussi ma familles et mes amis pour m’avoir supporté tout au long de mes études ; en particulier ma grande sœur pour ces précieux conseils.

Et enfin, j'exprime ma gratitude aux membres de mon jury de thèse ayant accepté d’évaluer mes travaux.

Abstract

Cars embed a steadily increasing number of Electric and Electronic Systems. The ISO 26262 standard discusses at length the requirements that these systems must follow in order to guaranty their functional safety.

One of the means at hand to ensure the automotive systems safety is to perform safety analyses. During these analyses, practitioners perform FTA and FMEDA in order to evaluate the “trust” that we have in a system. As big quantities of data are handled in those analyses, it would be of great help for them to have the possibility to efficiently generate a part of them and check their consistency.

This manuscript is the result of a thesis led on this subject. It focuses on the formalization of the data handled during the safety analyses in order to propose an efficient methodology for their generation. It presents the different works done, from the proposition of formal models for the safety related element behavior representation to the design and implementation of a process for consistent FMEDA generation based on Fault tree patterns.

Keywords

Functional Safety, Markov Chains, Automated Generation, Safety Mechanisms, Fault Trees, FTA, Coverage, AltaRica3.

Résumé

La complexité et la criticité des systèmes électroniques embarqués automobiles est en augmentation constante. Un nouveau standard concernant la sûreté de fonctionnement automobile (ISO 26262) permet d'établir un cadre et de définir des exigences sur les systèmes concernés afin de garantir leur sécurité.

Un des moyens permettant de vérifier la sûreté de ces systèmes consiste à effectuer des analyses dites de sureté de fonctionnement. Au cours de ces analyses, les praticiens effectuent des analyses de type FTA et FMEDA afin d'évaluer robustesse et la sûreté de ces systèmes. Lors de ces analyses, les praticiens manipulent une masse de données de plus en plus conséquente ; Ce qui a créé le besoin d'avoir un moyen de générer une partie de ces données efficacement et de vérifier leur cohérence.

Dans ce manuscrit, nous détaillons les travaux que nous avons effectués sur ce sujet, en nous concentrant principalement sur la formalisation des données manipulées durant les analyses de sûreté de fonctionnent afin de proposer une méthode efficace pour leur génération. Nous y présentons les différents travaux réalisés, de la proposition de modèles formels pour la représentation du comportement dysfonctionnel « d'élément lié à la sûreté » à la conception et mise en œuvre d'un processus pour la génération de FMEDA cohérentes à partir d'arbres de défaillances.

Mots-clés

Sûreté de fonctionnement, Chaines de Markov, Génération Automatique, Mécanismes de Sûreté, Arbres de défaillances, AdD, AltaRica3.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Chapter 1 Introduction

1.1 Context Presentation

Cars embed a steadily increasing number of Electric and Electronic Systems. Since the end of the 90's, automotive industry has changed its way to design vehicle and the underlying systems that compose these vehicles. Back then, the systems were designed following a federal architecture where one ECU was dedicated to one function or service.

The innovation pace have risen, particularly in electronics and computing which lead to replace mechanic and hydraulic commands by electronic components. Back then, each function of the car was developed independently from the others.

These embedded systems cover a large spectrum of the systems. Each system have now the following properties: A system fulfills several functions. And a function necessitates multiples systems to be fulfilled. Thus, systems are interconnected and communicate between each other.

The main advantage of this architecture is the reduction of the number of systems and ECU in the vehicle. However, it increases significantly the complexity of each of them.

With the growth of the complexity of the vehicles, the need to ensure their functional safety became more and more important. Thus, functional safety processes started to be implemented and followed by the automotive actors (constructors, tier 1…).

In 2011, ISO26262 standard was published (ISO 26262, 2011). This standard defines a number of constraints and rules that the development of automotive electric and electronic systems must obey in order to ensure their functional safety.

Since then, all the automotive industry actors must follow the requirement of this standard in order to produce "safe" cars.

1.2 Thesis subject presentation

The main objective of this thesis was to assess the functional safety process at Valeo and propose a solution for the generation of safety analyses. By analyzing it, our goal was to define the key points to work on in order to ensure the compliance to ISO26262, and define an efficient way to simplify the safety analyses and their generation.

In the following chapters, we first give an overview of the state of practices for the automotive functional safety: we first present the various activities composing the safety process; we give a fast study of the state of the art for the safety analyses generation and defend our research plan.

Following this, we present the result of our study on the automotive safety related elements: we focus on the safety mechanisms, study their failure behavior by representing them with the help of Markov chains and define the importance of the parameters characterizing them.

Next, we define and test fault tree patterns that represents these safety mechanism efficiently: based on these patterns, we define processes for ISO26262 specific developments (like metrics calculation) and FMEDA generation/check.

And finally, we provide high level models for the representation of automotive safety mechanisms: we define classes for each type of safety mechanisms based on known examples.

CHAPTER 2
AUTOMOTIVE SAFETY STATE OF PRACTICES

Chapter 2 Automotive Safety : State of Practices

2.1 Automotive Systems Safety & ISO 26262

Since the beginning of the 21th century, the integration of E/E systems in automotive vehicles has started to rise up the problem of multi-critical systems. Indeed, developed systems integrate both critical and non-critical functions. A function is considered as critical if it could lead to an Undesired Event (which causes an accident).

Moreover, many actors are involved in the development process of a car: car manufacturer and several suppliers (Tier 1, Tier 2…) which develop the products of the system defined by the OEM. Each company has its own development process; therefore it is necessary to define and follow robust design rules with documents and processes ensuring traceability.

Before 2011, as there were no directives on functional safety in the automotive industry, only a few companies decided to adhere (voluntary) to the state of the art defined in the IEC 61508 (IEC 61508, 2010).

IEC 61508 focuses on the overall development process of a system and the steps that have to be respected in order to achieve functional safety of electrical components. Particularly, it defines achievable goals for the specification, the design, the implementation and assessment of electrical/electronic programmable systems.

Since 2011, a derived version called ISO 26262 (ISO 26262, 2011) is used. This Standard is the result of the work between the major companies of the automotive domain in order to specify best practices for the documentation, the interactions between actors and the methods and techniques to justify the functional safety of automotive systems. This facilitates exchanges between OEMs and Suppliers by giving requirements to achieve.

Safety is divided into non-functional safety and functional safety:

- Functional safety addresses possible hazards caused by malfunctioning behavior of E/E systems including interaction of these systems. Typical examples of functional hazards are: steering column lock, engine racing and loss of front lighting.
- Undesired events such as electric shock, fire, smoke, heat, radiation, toxicity, flammability, reactivity, corrosion, release of energy, are considered as non-functional unless directly caused by malfunctioning behavior of E/E safety-related systems.

Technical measures considered in a design to cope with non-functional safety UEs are generally only based on fault avoidance (suppression of potential root causes).

Technical solutions to cope with functional safety UEs are based on fault avoidance and fault tolerance (avoid faults propagation).

The scope of ISO 26262 is on functional safety of automotive E/E systems. The standard defines functional safety as “absence of unreasonable risks due to hazards caused by malfunctioning behavior of E/E systems”

The ISO 26262 is divided in ten parts described in Figure 2:1.

Our work deals with the Part 4 and Part 5 which give all the safety requirements for the development of hardware automotive system. However, other parts are also very helpful for the understanding of these requirements and their application especially Part 10.

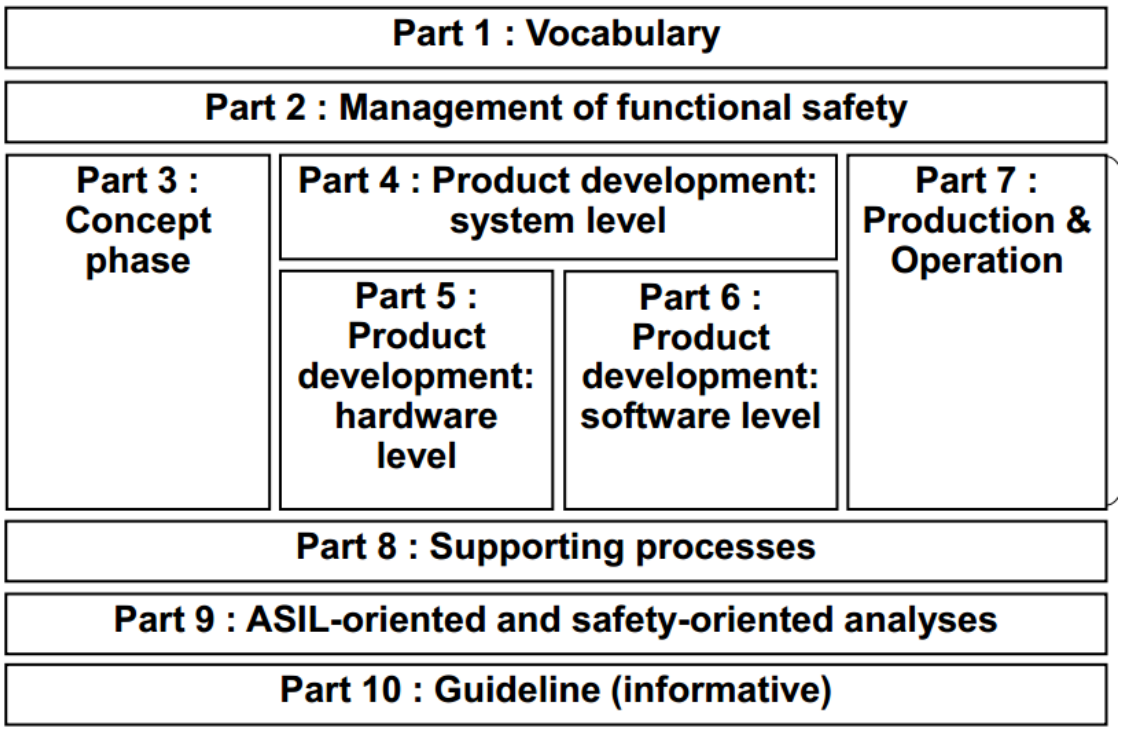

Figure 2:1 The Ten Parts of the ISO 26262 (ISO 26262, 2011)

2.2 Basic Concepts of Dependability & ISO 26262

Dependability is a key concept of any critical system. It could be seen as the aptitude to avoid the failures that occur during a service delivering. This service corresponds to the behavior perceived by the users (human or not) in interaction with the service.

Dependability is a well-documented concept, on which has been defined a complete taxonomy (Avizienis, et al., 2004). Indeed, dependability is defined by 6 main attributes, three treats and four categories of means.

2.2.1 From Dependability Attributes to Automotive Safety Integrity Levels

2.2.1.1 Dependability Attributes

In order to characterize the quality of a delivered service, dependability takes in the following attributes:

- **Availability**: readiness for correct service;
- **Reliability**: continuity of correct service;
- **Safety**: absence of catastrophic consequences on the user(s) and the environment;
- **Confidentiality**: absence of unauthorized disclosure of information;
- **Integrity**: absence of improper system alterations;
- **Maintainability**: ability to undergo modifications and repairs.

Depending on the industrial field, the significance of each attribute varies. This choice is based on the objectives that should be achieved by the developed service. For example, in transportation fields, reliability and safety are of prime priority; although, the rise of connected vehicles challenges increases the confidentiality importance.

In other fields, like communications, prime priority is given availability, reliability. Particularly, automotive systems are mainly focused on safety, availability and reliability attributes.

2.2.1.2 Automotive Safety-Integrity Level

In ISO 26262 Standard, a functional Undesired Event (UE) is rated according to its criticality on a five level scale (QM, ASIL A, ASIL B, ASIL C and ASIL D). The least critical effects are rated QM (Quality Management) and no specific safety requirement are associated to it in the standard. The most critical effects are rated ASIL D. A system functional UE with an ASIL is also called a hazard.

When assigning these levels, three parameters must be taken into account, see:

- Severity: Based on the severity of the potential injured or killed persons in the incident or accident (S1: Light and moderate injuries, S2: Severe and life-threatening injuries (survival probable) S3 Life-threatening injuries (survival uncertain), fatal injuries);
- Probability of exposure: Occurrence of the use case: E1: very low probability, E2: Low probability E3: Medium probability, E4: High probability;
- Controllability: It is a subjective concept that is based on the abilities of the driver to handle the hazard (C1: Simply controllable, C2: Normally controllable, C3: Difficult to control or uncontrollable).

The objective of these levels is to characterize the “safety” the system should be designed to ensure its functions correctly. The more the system is safety critical, the more the ASIL is high and the efforts required by the norm are stringent.

Table 2:1 Definition of the Safety-ASIL Matrix (ISO 26262, 2011)

		Controllability		
Severity of the harm	**Probability of exposure**	**C1**	**C2**	**C3**
S1	**E1**	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
S2	**E1**	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
S3	**E1**	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

Table 2:1 shows the relation between the Automotive Safety Integrity Levels (ASILs) and their defining parameters.

In the next section, we will present how ISO 26262 is taken into account in the Valeo safety process.

2.3 Valeo Safety Methodology

In the scope of ISO26262 standard deployment, the Valeo safety standardization working group created a new process dedicated to functional safety (Leeman, 2013).

It can be considered as an instantiation of ISO26262 requirements in order to simplify their integration in the overall Valeo conception/development process for E/E systems.

Figure 2:2 shows the activities belonging to the safety process in green and the other activities in blue. The safety process mainly covers the system activities (according to the ISO definition).

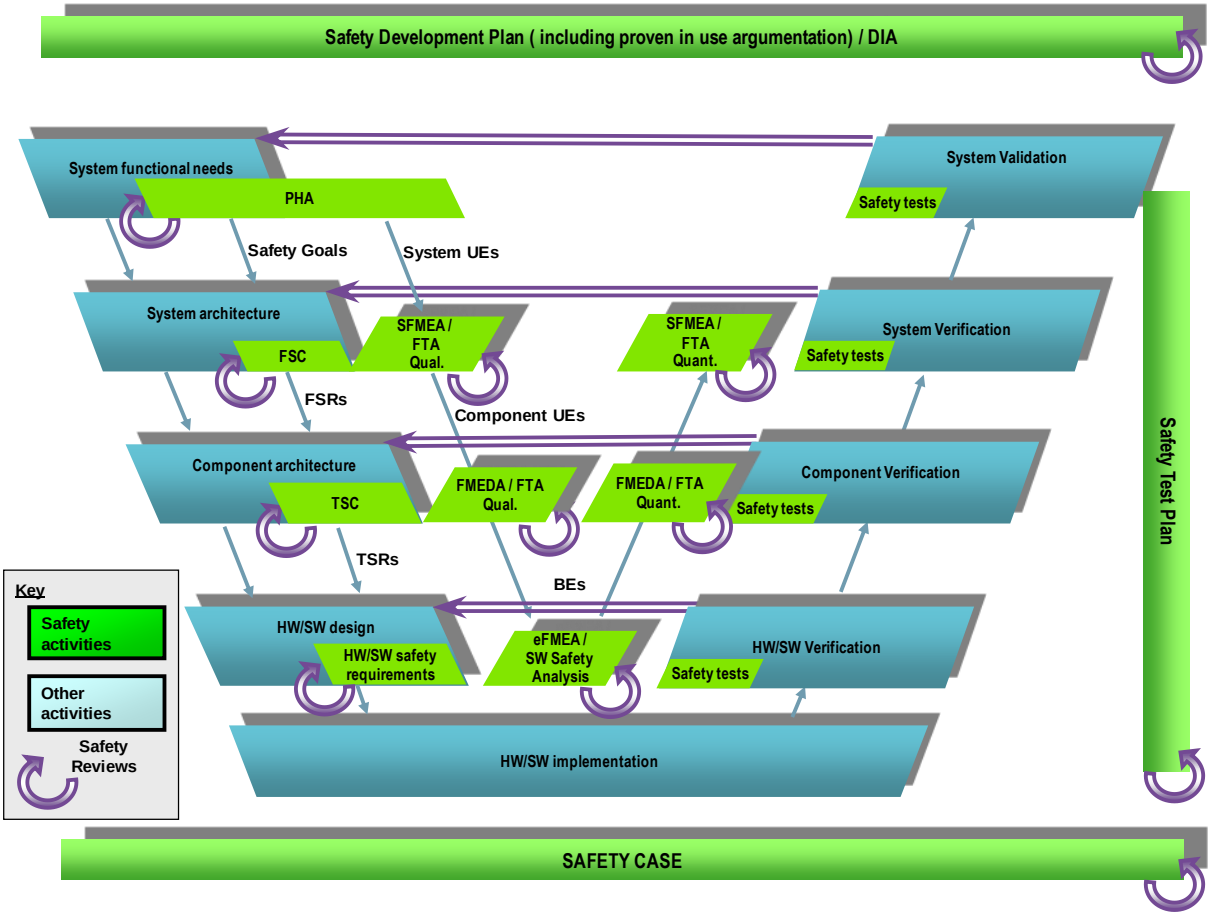

Figure 2:2 Overall safety process description

It should be noted that the Valeo definition of a system fits rather with the item as defined in ISO26262 Standard. In the rest of this document we cope with this definition.

The Preliminary Hazard Analysis (PHA) activity covers the Hazard Assessment and Risk Analyses (HA&RA) requirements of the Standard. A central new activity consists in designing safety concepts:

- the Functional Safety Concept (FSC) defined at system level,

- and the Technical Safety Concept (TSC) defined at component level.

If we simplify a little bit, we can say that the safety concept drafting is supported by the qualitative safety analyses and verified with the quantitative safety analyses. This is the reason why the first ones are represented on the left side of the V cycle and the others are on the right side. The safety tests belong to the existing tests activities, but the safety test plan belongs to the safety process. This safety test plan aims at test coverage verification. The safety reviews plus the safety aspects of the SW and HW processes verified by the HW and SW reviews cover the ISO requirements concerning the technical reviews, confirmation measures (including the safety assessment) as well as the safety audit.

The safety management is supported by a Safety Development Plan and one or more DIA (Development Interface Agreement). A DIA defines the respective responsibilities of Valeo and an external partner (customer, supplier or co-supplier) and the safety plan details how the Valeo activities will be performed on a project. The safety case gathers all the safety work-products (including HW and SW work-products). Its structure fits with the safety plan.

2.3.1 Safety Analyses

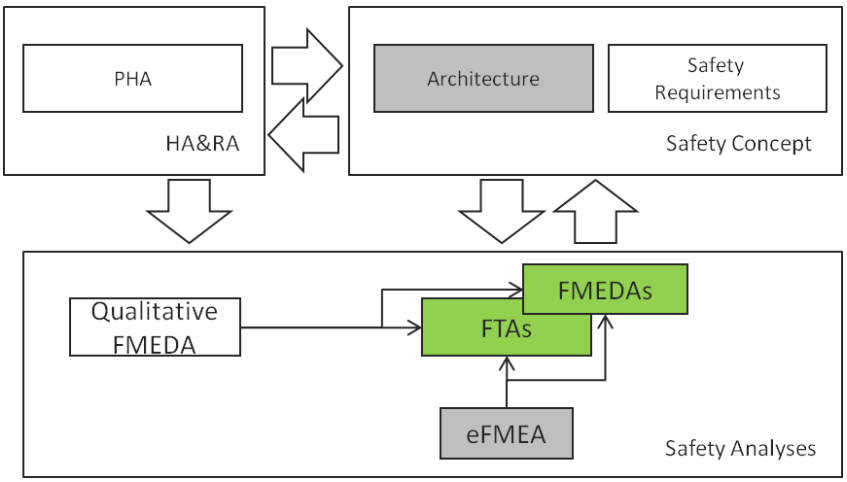

Figure 2:3 Safety analyses activities overview

Figure 2:3 gives an overview of safety analyses activities and their inputs/outputs. Preliminary Hazards Analyses (PHA) is performed for innovative projects where new functions are introduced. For more mature products, it is preferable to rather rely on the customers' requirements when they exist and are not too dissimilar.

There is one safety FMEA per architectural level. At item level, the System FMEA (SFMEA) is a qualitative analysis to show sufficient fault tolerance of the system as well as to support the system design and more particularly the FSC drafting. Its principle is to identify all the critical failure modes of the components composing the system and the way they propagate to cause the critical failure modes of it. These failure modes leading to undesirable events identified in the PHA.

Qualitative FMEDA is the equivalent of SFMEA at product level. It supports the drafting of the TSC. Its aim is to identify all the critical failure modes of the internal HW functional blocks of a component and the way they propagate to cause the system UEs identified in the SFMEA in order to define adequate safety mechanisms at component level.

When required, allocation of quantitative requirements to the components of the system is the first step for the calculation of the architectural metrics. It uses the components UEs identified in SFMEA and system safety mechanisms defined in the FSC as main inputs.

eFMEA analyses the HW schematics. It ensures an exhaustive identification of the HW functional blocks failure modes (the Basic Events) and allows calculating the failure rates of these Basic Events. It uses the HW parts failure rates and failure modes as inputs and is generally derivated from the architecture.

Quantitative FMEDA verifies the quantitative requirements allocated to a component for a particular component UE, using failure rates of the basic events calculated in eFMEA. For a given product, all the quantitative FMEDAs are derived from the Qualitative FMEDA. The consolidation of all components quantitative results is done at system level to verify that the architectural metrics targets are met.

In parallel to these bottom/up analyses, ISO26262 requires to perform top/down analyses. For a given functional UE, a fault tree is built to analyze a particular component failure mode. Fault Tree Analyses (FTA) analyzes causes as well as combination of causes of a particular component/system failure mode. FTA is also a powerful tool to define safety mechanisms and is more appropriate than FMEDA to identify independence requirements between architectural elements. It allows quantification of the residual failure rate of a component failure mode in addition to the computation of its failure probability and PMHF.

The quantity of data handled during these analyses is really consequent, and it would be of a great help for practitioners to have assistance when performing them. One of the main objectives behind this thesis was to bring simplification and consistency to these automotive safety analyses by investigating their automation possibility.

In the next section, we give an overview of the state of the art for generation of high level safety analyses (FTA and FMEDAs).

2.4 State of the Art

The automation of safety analyses generation problematic is not particularly new. Indeed, first works on this subject can be found in the beginning of the 90's and the widespread of the information systems.

Nowadays, fault trees (dynamic or not) can be generated rather easily from formal models (Perrot, et al., 2010). This is mainly due to the mathematical logic format behind their representation. In opposition to this, Failure Modes and Effects Analyses generation is rather difficult. This is mainly due to the content of these tables. Indeed, it contains mostly humanly written and understandable sentences, making them hardly extractable from simple models. This is the reason why we focus on FMEA generation in this section.

To begin, the first works dealing with the FMEA generation in the automotive industry date back to the *Flame System* (Price, et al., 1995) in the early 90's, it presented a system/tool that allows the FMEA generation from models of component and their possible faults by using the system description to extract the failure modes and their possible effects details.

In parallel, Montgomery, introduced the FMEA Streamlining tool based on analogic circuits simulator which simulated each failure mode on a circuit allowing the identification of the failures that impact the system (Montgomery & Pugh, 1996) (Montgomery & Marko, 1997).

Since then, various approaches where proposed. For example: Teoh & Case proposed FMEA generation through the use of functional diagrams starting from the design phase (Teoh & Case, 2004), Elqmvist introduced an FMEA generation method through a compositional approach based on component models. It uses safety interfaces to formally describe component failures (Elmqvist & Nadm-Tehrani, 2007) (Elmqvist & Nadm-Tehrani, 2008), Paige & Rose introduced a formalism using fault propagation and transformation analyses and allowed the deduction of the failure behavior of a system from the failure behavior of its components (Page & Rose, 2009). There are also many other approaches not really related to the safety, for example, Wang & pan, proposed an automatic process FMEA technique using the Little-JIL language (Wang & Pan, 2010).

Also, as the calculation power of the computers has grown up, the generation of the safety analyses from models has started to be more and more interesting and possible. Nowadays, we can identify 3 main approaches when it comes to the FMEA generation. We give an overview of these in the next sub-sections.

2.4.1 FMEA generation from functional models

This approach focuses on the extension of functional models with safety related data. However, previous experiences have proven that simulating the functional and failure behavior of a system needs a huge computing time which exponentially increases when the size of the system grows. Various works propose methods to overcome this problematic.

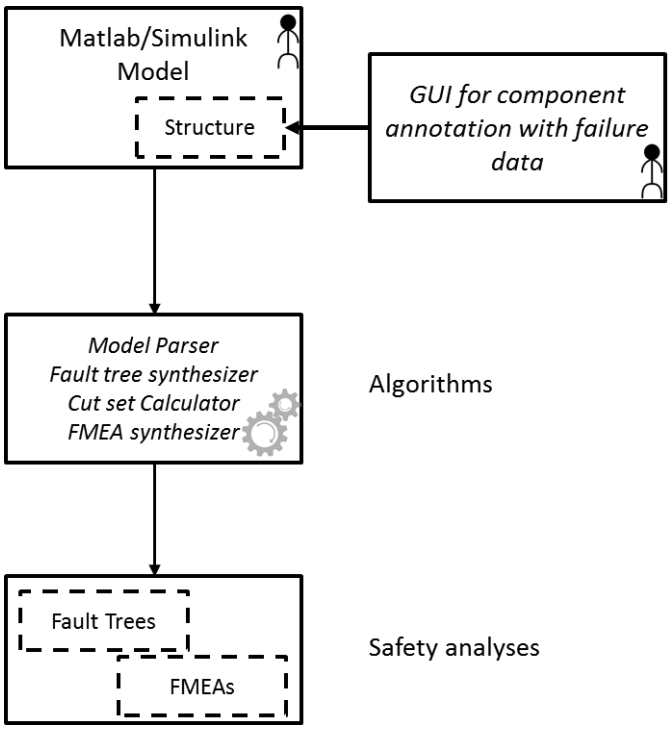

Figure 2:4 Simplified HiP-HOPS process overview

For example, Papadopoulos proposes methods and tools (HiP-HOPS) which consist in adding local failure data to Simulink diagrams at various levels, then, using the structure of these diagrams to propagate these failure data through the model. This information is then used to build fault trees. The fault trees are then converted into tables containing: The parts failures, their direct effects on the system and the effect that

they can cause by combining them. This table is finally used to generate Single point fault FMEA (critical), and multiple point fault FMEA (Papadopoulos & Parker, 2004) (Papadopoulos & Parker, 2005).

Figure 2:4 gives a simplified overview of the HiP-HOPS process for safety analyses generation.

2.4.2 FMEA generation from architectural models

This approach focuses on the extension of architectural model describing the systems functionality with safety related data.

For example, Idasiak & David with MéDISIS (Idasiak & David, 2008) proposes a method that consists in the use of SysML for the description of the functional behavior of a system by describing its architecture, its component hierarchy and the various dataflow that circulate in the system, then, combining these information with a database containing component failure modes to generate FMEA.

The generated FMEA can be analyzed and corrected. The corrections are taken into account to feed the database.

Figure 2:5 displays a simplified of the MéDISIS process for FMEA generation.

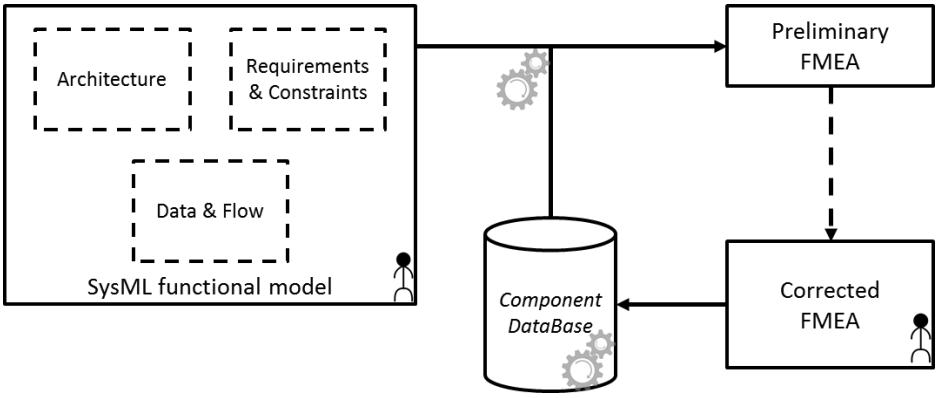

Figure 2:5 Simplified MéDISIS process overview

2.4.3 FMEA generation based on safety models

This approach focuses on the FMEA generation from dedicated models which describe the failure behavior of a system.

For example, Arbaretier & Brik presents how to generate FMEA using SimFia, a tool based on the AltaRica language which is dedicated to the description and simulation of failure behaviors. The tool allows to model systems' failures through different graphical abstraction views (application, physic and logic) which are associated to AltaRica code, which is then used to generate FMEA and other analyses using implemented algorithms (Arberetier & Brik, 2010).

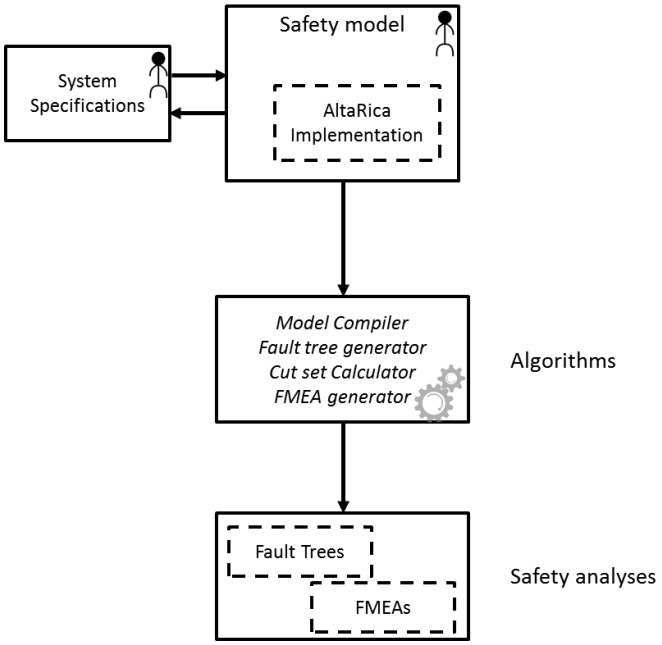

Figure 2:6 Simplified SimFia process overview

As another formalism following this kind of approach, we can mention Figaro which is another language for system failure modelling. (Torrente & Bouissou, 2008).

2.4.4 Discussion

We presented in this section a brief history for the safety analyses automatic generation; we illustrated the three main axes of research that are currently being investigated. Each one of these axes is indeed of a great interest for the evolution and simplification of the overall safety analyses process.

However, one of the objectives that we had when we started this thesis was to provide directly deployable and usable solutions. Indeed, in order to be able to deploy either of presented methodologies, there were two things that must be heavily considerate:

- Verify that the selected method(s) and tool(s) cope with the ISO26262 requirements and Valeo processes, and adapt them if not,
- Form and train the practitioners (automotive safety engineers), allowing them to efficiently use the provided tool.

This is why, we decided to provide a custom approach based on lower level formalizations and modelling. This will be explained in the next section.

2.5 Thesis Approach

One of the main objectives that we had to consider during this project was to provide Valeo with concrete solutions for enhancing their safety analyses process.

The first step in the realization of this objective was to assess Valeo safety analyses methodology and ISO 26262, the data that are handled during these analyses and the critical points were we could bring and apply our know-how.

The first critical point that was identified was the necessity to build formal definition for understanding the behavior of the elements which are assessed during the safety analyses. To solve this, we built state transition diagrams and Markov chains to represent the failure behavior of these elements (hardware blocs and automotive safety mechanisms). Based on those diagrams we performed a study in order to determine the impact of each parameter characterizing these elements. This is the scope of the Chapter 3 of this manuscript.

Following this, in order to exploit those results and formalize our safety analyses, we tested various fault tree patterns. Each pattern was compared with the Markov chains previously defined in order to challenge its accuracy in extreme cases. This is the scope of the chapter 4 of this manuscript.

Then, based on those fault trees patterns, we presented some specific ISO26262 developments that were realized. We first introduce a fault tree based methodology for the ISO26262 architectural metrics calculations and its implementation, then, we present the work that we performed to generate quantitative FMEDA from fault trees and to verify the coherency between a qualitative FMEDA and its fault tree. This is the scope of the chapter 5 of this manuscript.

To finish, we present in the chapter 6, some of the side works that we performed in order to model the failure behavior of safety related elements using a high level modeling language (AltaRica 3)

CHAPTER 3

SETTING THE FOUNDATION

SAFETY RELATED ELEMENTS

Chapter 3 Setting the Foundation: Safety related Elements Behavior

The ISO 26262 (ISO 26262, 2011) standard discusses at length the use of Safety Mechanisms and how to estimate their contribution to functional safety. To do so, it relies essentially on Fault Tree models or ad-hoc formula. Such models or formulas are indeed of interest for practitioners. But they are only approximations. Without a more explicit representation of failure scenarios to serve as a reference, it is hard to check them for completeness, to understand their domain of validity and to ensure their accuracy. Explicit models have been proposed by several authors for Safety Instrumented System described in IEC 61508 Standard(Commission, 1998) (see e.g.(Innal, et al., July 2010),(Jin, et al., 2011)). In the case of the ISO 26262 standard, at least to our knowledge, this work has not been done yet.

The purpose of this chapter is therefore to fill this hole by proposing generic Markov models for Electric and Electronic Systems reinforced by first order and possibly second order Safety Mechanisms. The interest of these models is twofold: first, they are of a great help to clarify the behavior of safety mechanisms; second, they make it possible to determine the domain of validity of simpler models such as Fault Trees or ad-hoc formulas of the standard.

The remainder of this chapter is organized as follows. First, we present two typical examples of safety mechanisms in Section 3.1. Then, we propose Markov models for these safety mechanisms in Section 3.2. We report numerical results obtained on these models in Section 3.3 and we discuss their significance. Finally, we review related works in Section 4.4.

3.1 Two Typical Examples of Safety Mechanisms

In this section, we introduce two representative examples of automotive systems embedding safety mechanisms.

3.1.1 Vehicle Management Unit for Inversion

We shall first consider the case of a Vehicle Management Unit (VMU). In an electric vehicle, a VMU is responsible for commanding the electric motor inverter, among other functions. A VMU consists typically in a microcontroller which, given certain inputs (gas and brake pedal positions), sends a torque set-point to the inverter that in turn commands the electric motor (traction and regenerative braking), as illustrated Figure 3:1.

Such a VMU is a critical function: if the microcontroller gets stuck in a loop and continuously sends a command higher (or lower) than expected, it could lead to unintended vehicle acceleration or braking.

In order to prevent such hazards, a watchdog is added which is in charge of bringing the system to a safe state in case the microcontroller is detected to be stuck. The watchdog is an electronic component that is used to detect and recover from microcontroller malfunctions. The microcontroller refreshes regularly the

watchdog in order to prevent him from timing out. If it gets stuck in a loop, the watchdog cannot be reset, so the watchdog times out and sends a reboot order to the microcontroller.

Such a watchdog is a first order safety mechanism based on error detection.

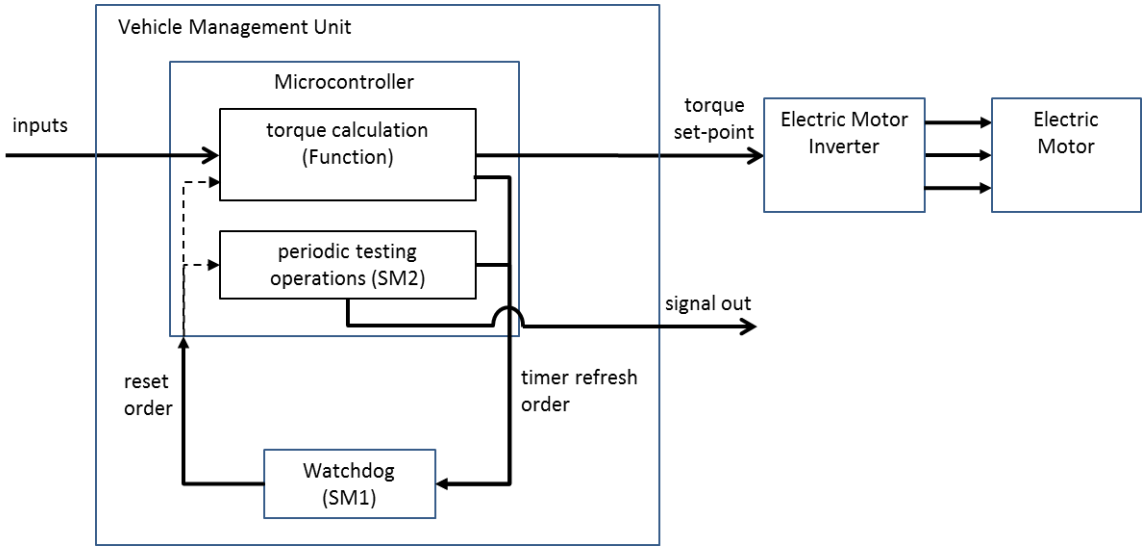

Figure 3:1 Simplified functional representation of the Vehicle Management Unit for Inversion

As a physical component, the watchdog may fail (although the reliability of the watchdog is much higher than the one of the microcontroller). Also, the watchdog is able to detect only certain kind of errors of the microcontroller: typically, it is not able to detect memory corruption problems.

In order to ensure that the watchdog is working, the microcontroller tests the watchdog at each vehicle start. The role of this second order mechanism is to warn the driver in a case of a problem with the watchdog. It may itself fail and is itself not able to catch all of the problems of the watchdog.

As the torque calculation function and the second order safety mechanism function are never executed in parallel, their failures are considered as independent (and are independent from watchdog failures).

The above example is representative of safety mechanisms based on error detection as embedded for instance in electric steering column controller, electric braking, several types of microcontrollers protected with watchdogs and more generally command-control systems.

3.1.2 Electric Driver Seat Controls

Another type of safety mechanism is used in Electric Driver Seat Controls (EDSC). An EDSC allows the driver to tune his seat position. A spurious tuning action while the vehicle is running (over a certain speed, e.g. 10km/h) can indeed cause an accident, for instance because the driver is no longer able to reach the brake pedal or because he gets suddenly pushed onto the steering wheel.

In order to prevent this from happening, the system embeds a mechanism in charge of turning off the power supply of the EDSC when the vehicle is running. This first order mechanism is therefore based on

inhibition. As previously, it is in general completed with a second order one in charge of testing it at each vehicle start (obviously, it cannot be tested while the vehicle is running).

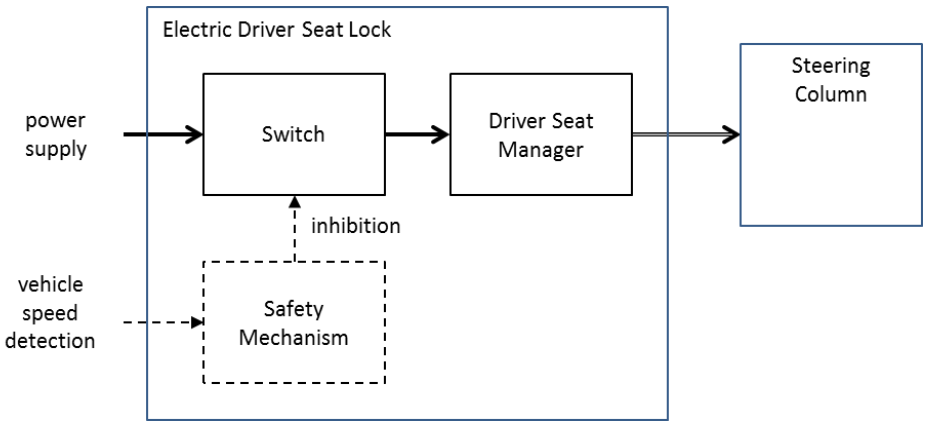

Figure 3:2 Functional representation of an Electric Driver Seat Control

The above mechanism is representative of safety mechanisms based on inhibition, as embedded for instance in Electric Steering Column Lock, Automatic Doors opening systems and more generally all systems that must be inhibited when the speed of the vehicle gets above a give threshold.

3.1.3 Discussion

The implementation of the safety mechanisms presented in this section is a practical way to enhance the automotive systems safety without expensive physical redundancy. These are used in order to reach the Probabilistic Metric for random Hardware Failures (PMHF) target.

The majority of automotive first order safety mechanisms can be actually categorized in either of the two categories presented above:

- Most of them are based on error detection. The idea is to switch the system into a safe state when an error is detected. These safety mechanisms are usually made of two elements: the detection device and the actuation device.
- Some of them inhibit the system they protect when the vehicle is in a state where the failure of the system is potentially dangerous.

As the unavailability of a first order safety mechanism has in general no direct influence on the system it controls, it can hardly be perceived by the driver. A second order safety mechanism is thus often added in order to check periodically the availability of the first one, typically when the engine is turned on or the vehicle starts to move. The role of such a second order mechanism is to warn the driver.

3.2 Generic Markov Models

To have a clear understanding of the behavior of Electric and Electronic Systems in presence of failures (including those of safety mechanisms), the best method is probably to design state/transition models for

these systems. It is often the case that Markovian hypotheses are verified or at least are a good approximation for calculation purposes so that these models can be turned into Markov chains in a straightforward way.

In this section, we shall propose Markov chains for systems of each of the two above categories. These Markov chains are generic in the sense that one has just to adjust values of parameters (such as failure rates, coverage rates…) to assess the safety of a particular system. Markov chains presented hereafter can be subsequently embedded into larger Markov models or approximated either by means of Fault Tree constructs or by ad-hoc formulas. They serve as a reference.

3.2.1 Case of a Hardware Block protected by a First Order Safety Mechanism Based on Error Detection

Let us consider first the case of a Hardware block HB protected by a first order safety mechanism SM1 based on error detection. The generic Markov chain for this system is given in Figure 3:3.

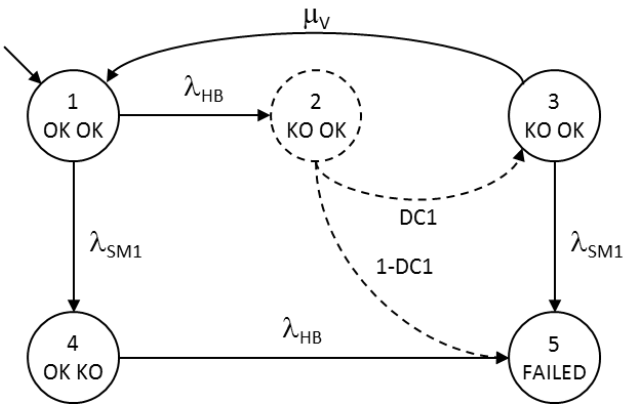

Figure 3:3 Generic Markov chain for a Hardware Block protected by a first order Safety Mechanism based on error detection.

Such a system fails in a dangerous state if both the hardware block and the safety mechanism fail, no matter in which order. Therefore, the Markov chain encodes basically three failure scenarios.

In the initial state (1), both the hardware block and the safety mechanism are working. The failure rates λ_{HB} for the hardware block and λ_{SM1} are assumed to be constant over the time (no ageing effect). If the hardware block fails first, the system goes to state 2, where the safety mechanism detects or not this failure instantaneously. As a graphical convention, we denote instantaneous states and their outgoing probabilities by dashed lines, as on the figure. The probability not to detect the failure is 1-DC1, where DC1 stands for the diagnostic coverage of the safety mechanism. In the state (2), if the failure of the hard block is not detected the system goes to the failure state (5) (first failure scenario). Otherwise, it goes to the safe state (3). In this state, the mean time before the vehicle is taken to the garage is T_M, i.e. the repair rate of the hardware block is $\mu_V = 1/T_M$. Now, if the safety mechanism fails before the vehicle is repaired, then the system goes to the failure state (5) (second failure scenario). Otherwise it goes back to the initial state (1).

Finally, if, in the initial state, the safety mechanism fails before the hardware block fails, then the system goes to state (4). In this state, we have nothing to do but to wait until the hardware block fails to go into the failure state (5) (third failure scenario).

Note that since there is no mean to detect a failure of the safety mechanism, there is no mean to repair it neither. Moreover, we assume that neither the hardware block nor the safety mechanisms are inspected during periodic maintenances of the vehicle. This hypothesis is realistic, although pessimistic.

3.2.2 Case of a Hardware Block protected by First Order Mechanism based on Error Detection and a Second Order Safety Mechanism

We shall consider now the case of a hardware block HB protected with a first order safety mechanism SM1 based on error detection which is itself tested by a second order safety mechanism each time the vehicle starts. The generic Markov chain for such a system is given in Figure 3:5.

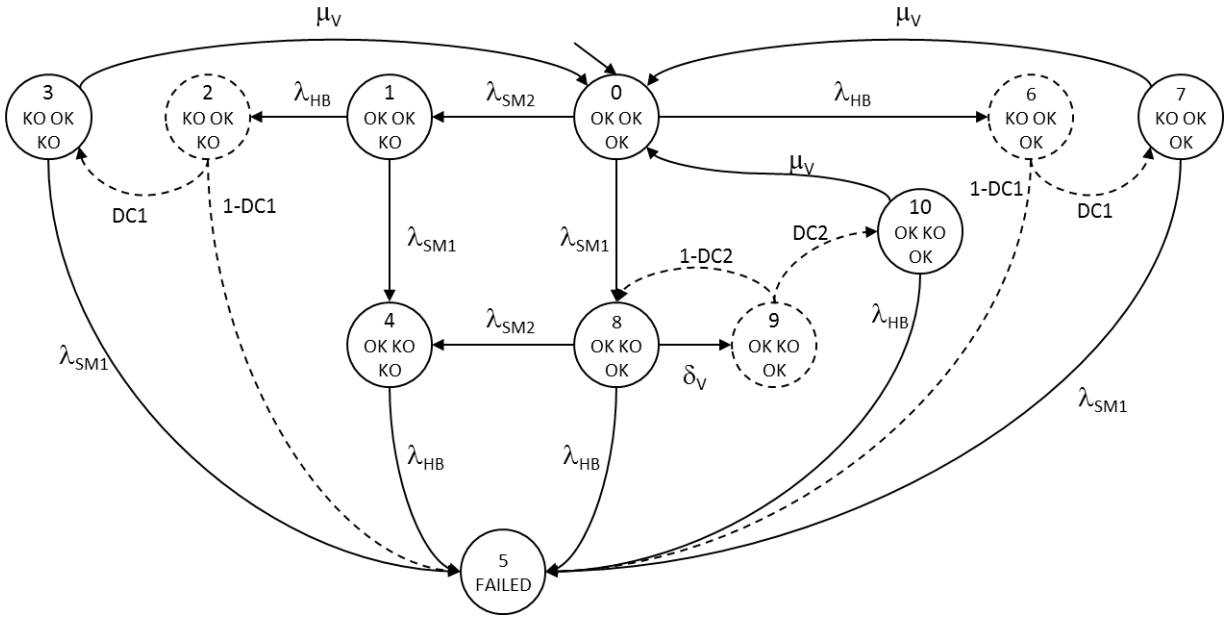

Figure 3:4 Generic Markov chain for a Hardware Block protected by a first order Safety Mechanism based on error detection and a second order Safety Mechanism.

This model extends the previous one. The second order mechanism has its own failure rate λ_{SM2} as well as its own diagnostic coverage DC2. Note that it is assumed that when the vehicle is taken to the garage, it is fully repaired and is as good as new after this repair.

In the initial state (0), the hardware block HB and the two safety mechanisms SM1 and SM2 are assumed to work correctly. Now there are three possibilities:

- The second order mechanism fails first. In that case, according to our hypotheses, we are exactly in the same situation as if there was no second order mechanism. So the model obeys the same pattern as previously. We kept actually the same numbering of states 1 to 5 to emphasize this point.
- The hardware block fails first. This situation is also very similar to the previous one, for the second order mechanism plays no specific role in the subsequent scenarios. State 0, 6 and 7 are therefore symmetric to states 1, 2 and 3. The only difference stands in the availability of the second order mechanism.

- The interesting scenarios are therefore those where the first safety mechanism fails first, i.e. the system goes to state 8. We shall now develop these scenarios.

In state 8, we are in the situation where the first order safety mechanism failure is unnoticed. Here again there is a race condition amongst three possibilities:

- The hardware block fails first, including before the current journey ends. In that case, the whole system fails (state 5).

- The second order safety mechanism fails first. In that case, we can make the pessimistic assumption that the driver did not notice the warning before this failure. So, we are back to the situation where there is no second order safety mechanism (and the first order one is failed), i.e. to state 4.

- The current journey ends before both the hardware block and the second order mechanism fail (state 9). We can assume that the mean time before the journey ends is Tj so that the transition rate between states 8 and 9 is $\delta_V = 1/T_J$. Now at the next start of the vehicle, the second order mechanism tests the first order one with a probability DC2 of successful detection. If the detection is successful (state 10) then either the driver takes the vehicle to the garage before the hardware block fails (in which case the system goes back to the initial state 0) or the hardware block fails first (in which case the whole system fails, i.e. goes to state 5). If the second order mechanism does not detect the failure of the first order one, then we have to wait for another start of the vehicle to make the test again (so the system goes back to state 8)

It is worth to note that the model described here is quite different from those proposed for Safety Instrumented Systems in references (Innal, et al., July 2010),(Jin, et al., 2011). The difference stands mainly in assumptions about the maintenance policy. As already pointed out, the designer of an automotive Electric and Electronic system has no control on maintenance. So, he has to make pessimistic hypotheses about what the driver will (reasonably) do.

3.2.3 Case of a Hardware Block protected by a First Order Safety Mechanism based on Inhibition and a Second Order Safety Mechanism.

We shall now consider the case of a hardware block HB protected with a first order safety mechanism SM1 that inhibits the hardware block functionality, itself periodically tested by a second order safety mechanism SM2. The generic Markov chain for such a system is given Figure 3:5. As the reader has immediately noticed, this model is embedded in the previous one. The reason is that if the hardware block fails before the first order safety mechanism, then there is nothing to inhibit and the system is safe (but of course not available).

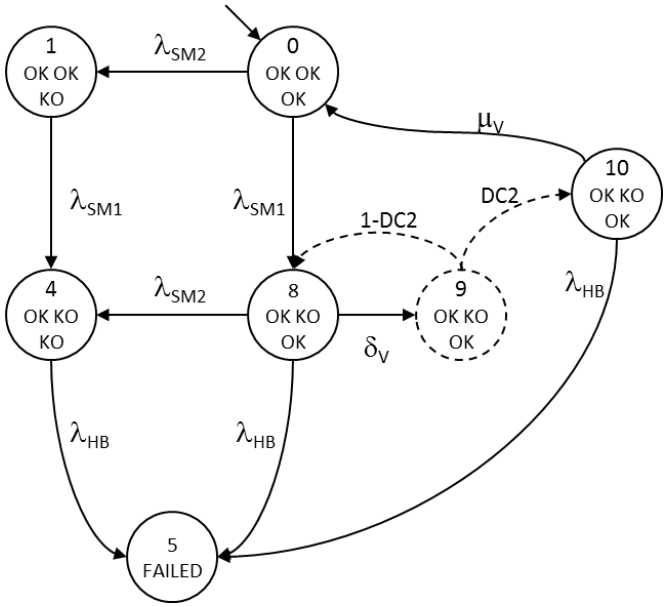

Figure 3:5 Generic Markov chain for a Hardware Block protected by a first order Safety Mechanism based on inhibition and a second order Safety Mechanism.

Note also that there is no detection device and therefore no diagnostic coverage for the first order safety mechanism.

From now on, this chapter will focus on studying the behavior of the first orders safety mechanisms based on detection and their second safety mechanism: As the markov model of the safety mechanisms based on inhibition seems to be a sub model of the model for the ones based on detection, every ascertainment made on these should be applicable to the ones based on inhibition.

3.3 Experimental Study for Detection Based Safety Mechanisms

Once the modeling of prototypical Electric and Electronic systems was established on the solid ground of the Markov chains presented in the previous section, we were in position to study the sensitivity of their safety to the variations of their reliability parameters. This section reports experiments we made on the model pictured inFigure 3:4, which is the most general one. To do so, we used the XMRK tool developed by one of the authors (Rauzy, 2004).

3.3.1 Realistic Values of the Parameters

In practice, mission times, transition rates and diagnostic coverages are by no means arbitrary. They vary within bounds from one system to the other but this variation is rather limited.

The considered lifetime of a vehicle is about 10000 driving hours. This corresponds to an average of 15 years or 400 thousand kilometers (660 hours of driving per year, with an average speed of 40km/h). We performed most of the calculations for this value.

The failure rate of hardware blocks (λ_{HB}) stands typically between 10^{-6} and 10^{-7} failures per hour. The failure rates of first and second order safety mechanisms (λ_{SM1} and λ_{SM2}) stand typically between 10^{-6} and 10^{-8}

failures per hour. We made most of the experiments around these values which corresponds to the failure rates ranges of the majority of the automotive components extracted from IEC 62380 [12]."

ISO 26262 annex D clarifies the evaluation of diagnostic coverage of safety mechanisms. Different tables are proposed in order to identify the type of safety mechanism that allows the detection of specific element failures. It also associates to each of those combinations the expected diagnostic coverage value, which represents the effectiveness of a safety mechanism with respect to the different failures modes(ISO 26262, 2011). The diagnostic coverage is typically sorted into three ranks: Low (60%), Medium (90%) and High (99%). However, these values can be adapted based on the analysis of the component or with the expert judgment in order to take into account specific characteristics such as specific implementations constraints or specific test periodicity. Also, a 100% diagnostic coverage can be considered if it can be justified. In practice, as it relies on the expert judgment, it's very unlikely to have a diagnostic coverage percentage with more than one or two decimal digits (e.g. 99.5%, 99.95%).

The mean journey time ($T_J = 1/\delta_V$) is of course more difficult to estimate. It is usually taken as to be 1 hour. We made it vary from this value to larger values to take into account a large variety of situations.

Similarly, the mean time before the vehicle is taken to the garage when a warning is raised ($T_M = 1/\mu_V$) depens dramatically on the driver. We made it vary also from 1 hour (i.e. the journey mean time) to the lifetime of the vehicle. Here again the ISO26262 standard provides typical values (Part 5, requirement 9.4.2.3, note 2) of the average time to vehicle repair, depending on the fault type:

- 200 vehicle trips for reduction of comfort features;
- 50 vehicle trips for reduction of driving support features;
- 20 vehicle trips for amber warning lights or impacts on driving behavior;
- One vehicle trip for red warning lights.

The time taken for repair is usually not considered (except to evaluate hazards that can expose maintenance personnel).

Table 3:2 summarizes realistic variations of the values of parameters.

Table 3:2. Typical Values of Parameters

	Lower bound	Higher bound
λ_{HB}	1E-07	1E-06
λ_{SM1}	1E-08	1E-06
DC1	0%	100%
$T_J = 1/\delta_V$	1	10

	Lower bound	Higher bound
λ_{SM2}	1E-08	1E-06
DC2	0%	100%
$T_M = 1/\mu_V$	1	10000

In the case of the Vehicle Management Unit presented Section 3.1.1, the per hour failure rate of the hardware block, i.e. the torque calculation part of the microcontroller has an estimated value of 0.4E-6. This estimation results from the weighting of failure probabilities and rates of different constituent of the microcontroller. The per hour failure rate of the watchdog is estimated at 5.0e-8. The diagnostic coverage of the watchdog is estimated from its capacity to detect different failure modes of the microcontroller and the proportion of failures of each mode. For a simple watchdog it would be around 60%, for a more elaborated

watchdog (so-called window watchdog) it would be around 90%. The per hour failure rate and diagnostic coverage of the second order mechanism are estimated respectively at 0.4E-6 and 60%.

3.3.2 Most Influential Parameters

According to numbers given in Table 3:2, the hardware block and both safety mechanisms are reliable with respect to the expected mission time of vehicle. As a consequence, scenarios involving more than one or two failures of these components are extremely improbable. Although the Markov chain pictured Figure 3:4 encodes an infinite number of failure sequences, only the shortest ones are of real interest. Figure 3:6 presents an unfolded (tree-like) view of this Markov chain. Sequences that go back to an already visited state are not expanded so to keep only shortest sequences.

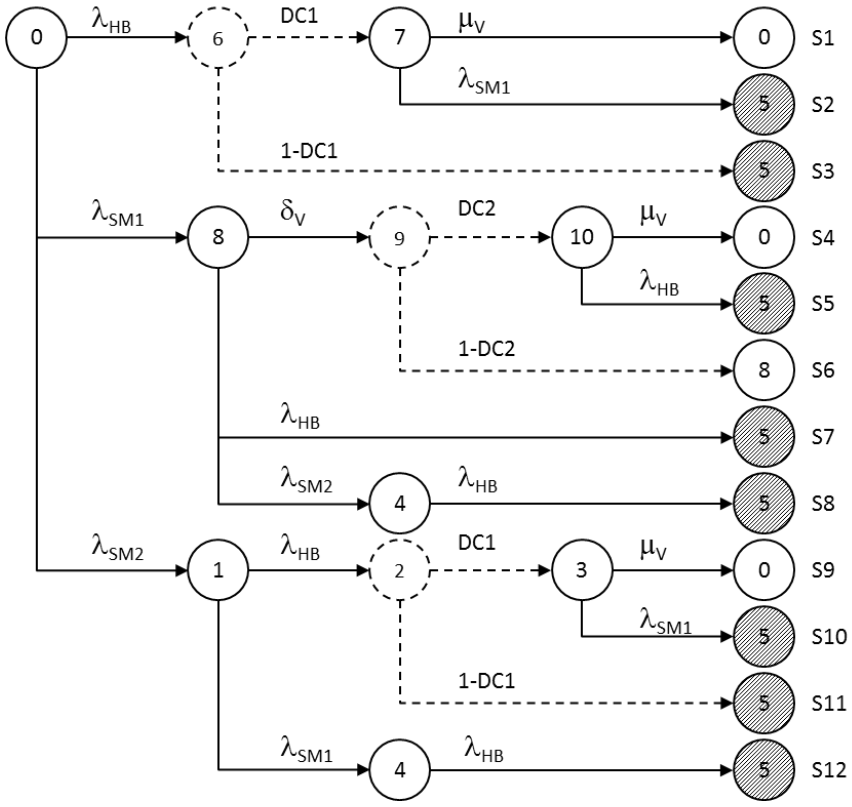

Figure 3:6 Unfolded view of the Markov chain representing hardware block protected with a first and second order mechanisms based on error detection.

Figure 3:6 makes clear that all of the failure sequences involve the failure of the hardware block. Therefore, its failure rate is an influential parameter. To illustrate this point, we calculated the probability of failure of the system for different values of λ_{HB} (λ_{HB} = 1.00E-6, 0.80E-6, 0.60E-6, 0.40E-6, and 0.20E-6 h^{-1}) and fixed values of the other parameters: λ_{SM1} = 1.00E-6 h^{-1}, DC1 = 99%, λ_{SM2} = 1.00E-6 h^{-1}, DC2 = 99%, T_J = 1 hour, and T_M = 10 hours. We made these calculations from 0 hour to 20000 hours by step of 100 hours. Values of the failure probability of the system are plotted Figure 3:7. This figure shows that the dependence of the failure probability w.r.t. the failure rate of the hardware block is quasi- linear. We observed such a quasi-linear dependence for other realistic values of the other parameters.

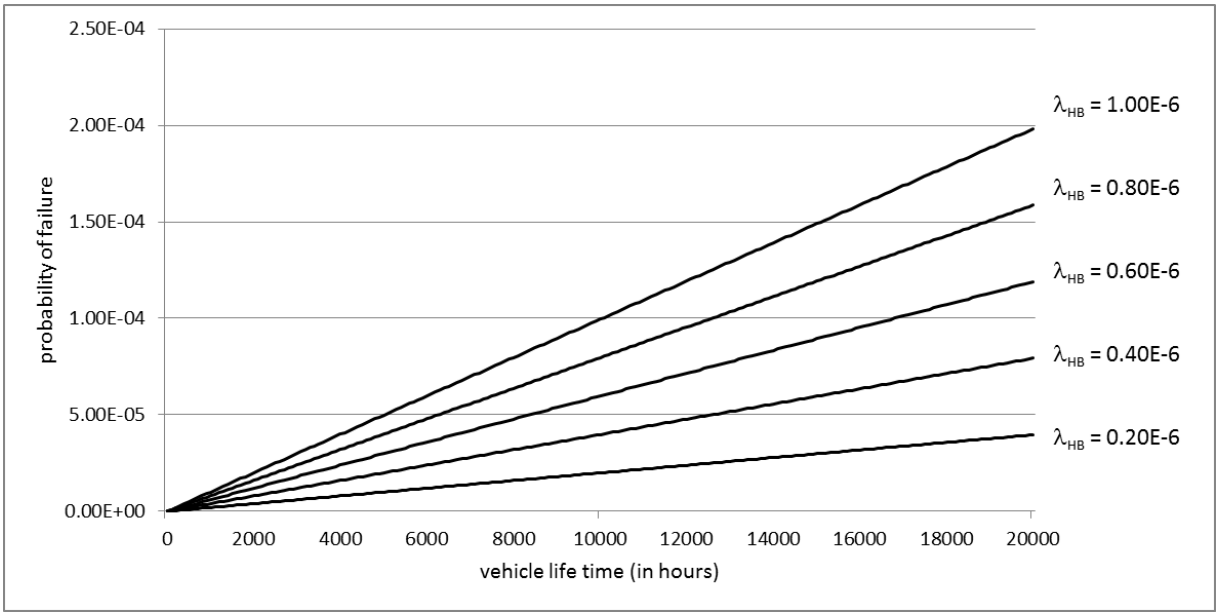

Figure 3:7 Variations, mutatis mutandis, of the failure probability with respect to the failure rate λ_{HB} of the hardware block (with λ_{SM1} = 1.00E-6 h^{-1}, DC1 = 99%, λ_{SM2} = 1.00E-6 h^{-1}, DC2 = 99%, T_J = 1h, T_M = 10h).

Table 3:3 gives the quotient of the failure probability by λ_{HB} for different times (and different values of λ_{HB}). These numbers confirm our quasi-linearity hypothesis.

Table 3:3 Quotient of the probability of failure divided by λ_{HB} for different mission times

		vehicle life time (in hours)			
		5000	10000	15000	20000
λ_{HB}	1.00E-6	49.89	99.54	148.97	198.19
	0.80E-6	49.89	99.54	148.97	198.19
	0.60E-6	49.89	99.54	148.97	198.19
	0.40E-6	49.89	99.54	148.97	198.19
	0.20E-6	49.89	99.54	148.97	198.19

We performed similar experiments to determine the influence of the diagnostic coverage DC1 of the first order mechanism on the failure probability. We let DC1 vary (DC1 = 95%, 96%, 97%, 98%, 99%) while the other parameters are fixed (λ_{HB} = 1.00E-6, λ_{SM1} = 1.00E-6, λ_{SM2} = 1.00E-6, DC2 = 99%, T_J = 1h, T_M = 10h) and we computed the failure probability from 0 hour to 20000 hours by step of 100 hours (although this value is twice higher than the vehicle lifetime, it allows a better visualization and interpretation of the different behaviors). Results are plotted Figure 3:8. This figure shows clearly the direct influence of this parameter on the failure probability. Again, similar results are obtained for different values of the other parameters.

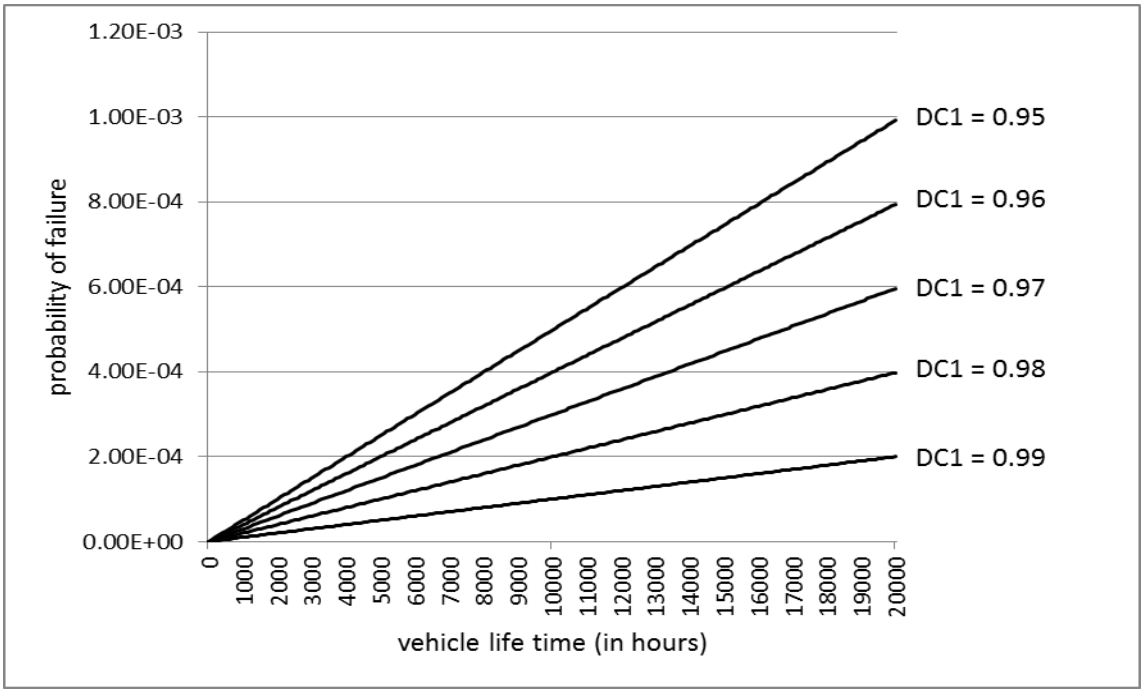

Figure 3:8 Variations, mutatis mutandis, of the failure probability with respect to the diagnostic coverage DC1 of the first order safety mechanism.

3.3.3 Influence of Other Parameters

We established so far that both the failure rate of the hardware block and the diagnostic coverage of the first order mechanism have a direct and quasi-linear influence on the failure probability of the whole system. What about the other parameters?

Figure 3:6 makes clear that sequences S9, S10, S11 and 12 are obtained respectively by prefixing sequences S1, S2, S3 and S7 with a failure of the second order mechanism (λ_{SM2}) and that sequence S8 is obtained from sequence S7 by inserting a failure of the second order mechanism in between the failure of the first order mechanism one and the failure of the hardware block. Moreover, the failure of the second order mechanism occurs only in sequences S8, S9, S10, S11 and S12.

As a consequence, the failure rate of the second order mechanism cannot greatly influence the probability of failure of the system.

There are two possibilities here: either we consider a perfect or nearly perfect diagnostic coverage of the first order mechanism, or we consider an imperfect (although possibly quite good) one.

If the diagnostic coverage is not perfect, it turns out that the other parameters have a minor role in the determination of the failure probability. The failure probability comes almost exclusively from the scenario: failure of hardware block (state 0 to state 6), non-detection of this failure (state 6 to state 5). As an illustration consider the four following extreme cases.

- Case 1: both the first order and the second safety mechanisms are highly reliable ($\lambda_{SM1} = \lambda_{SM2} =$ 1.0E-8), the second order mechanism detects perfectly the failures of the first one (DC2 = 100%) and the driver takes immediately the vehicle to the garage when she is advised to do so (T_M = 1h).

- Case 2: Similar to case 1, but with a passive driver who never takes the vehicle to the garage (T_M = 20000h).
- Case 3: the first order safety mechanism is only reasonably good (λ_{SM1} = 1.0E-6) and the second order mechanism is ineffective (DC2 = 0, T_J = 20000h). The driver is active (T_M = 1h).
- Case 4: similar to case 3, but with a passive driver (T_M = 20000h).

In all of the cases, we took λ_{HB} = 1.0E-6.

As shown by Figure 3:9, the probability of failure of the worst case (case 4) is less than three times as much as the probability of failure of the best one (case 1) after 20000 hours and only two times this probability after 10000 hours (the expected life time of a vehicle).

If the first order safety mechanism is highly reliable (λ_{SM1} = 1.0E-8), the driver behavior has not much influence. If it is only reasonably reliable (λ_{SM1} = 1.0E-6), the driver behavior has some influence.

With a smaller value diagnostic coverage of the first order safety mechanism, e.g. DC1 = 90%, the differences between the above test cases are negligible. The situation is rather different in the case of a perfect or nearly perfect diagnostic coverage of the first order mechanism. First, the failure rate of the first order mechanism comes into the play, second the behavior of the driver has a great influence especially in the case the first order mechanism is highly reliable, as illustrated Table 3:4.

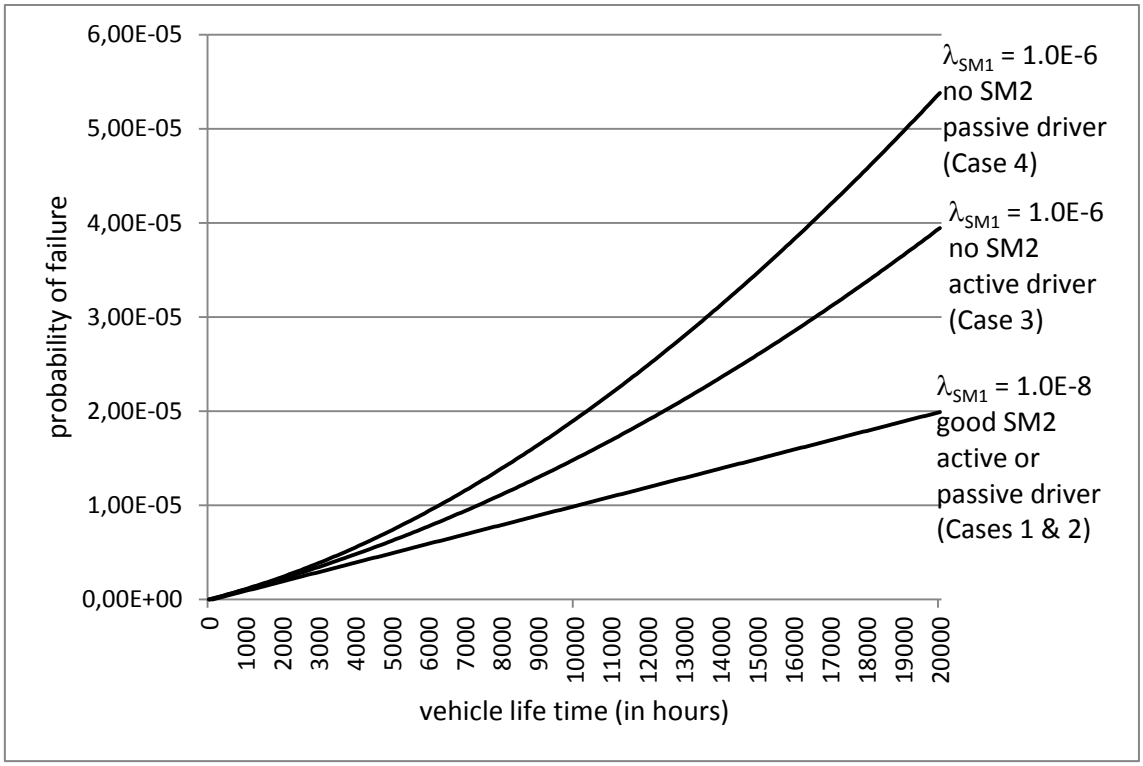

Figure 3:9 Influence of other parameters (but λ_{HB} = 1.0E-6) in case of an imperfect diagnostic coverage of the first order mechanism (DC1 = 99%).

Table 3:4 Influence of other parameters (but λ_{HB} = 1.0E-6) in case of a perfect diagnostic coverage of the first order mechanism (DC1 = 100%).

	vehicle life time (in hours)			
	5000	10000	15000	20000
λ_{SM1} = 1.0E-8, T_M = 1h	1.49E-11	3.13E-11	5.02E-11	7.28E-11
λ_{SM1} = 1.0E-8, T_M = 20000h	2.22E-08	8.36E-08	1.76E-07	2.92E-07
λ_{SM1} = 1.0E-6, T_M = 1h	1.25E-06	4.98E-06	1.12E-05	1.98E-05
λ_{SM1} = 1.0E-6, T_M = 20000h	2.40E-06	9.21E-06	2.00E-05	3.44E-05

3.3.4 Wrap-Up

The large experimental study we performed showed that, within the bounds set up by the current technologies, the two most influential reliability parameters are the failure rate of the hardware block and the diagnostic coverage of the first order safety mechanism. In a case of a perfect diagnostic coverage of the first order mechanism, the failure rate of the first order mechanism and the driver behavior have a significant impact on the reliability of the system. In all of the cases, the reliability of the second order mechanism has only a minor influence.

We can also tell that in the case of safety mechanisms based on inhibition, the most two influent parameters are the failure rate of the first order safety mechanism followed with the failure rate of the hardware block. Indeed, these safety mechanisms are not based on detection, so the diagnostic coverage has no influence on them. Also, as the hardware block cannot provoke a failure while inhibited, the only way to induce a component failure is that the safety mechanism fails first. For the the influence of the rest of the parameters, they are similar to the one analysed on the safety mechanisms based on detection whith perfect diagnostic coverage.

3.4 Related Works

As we said in the introduction, the design of Markov models for safety systems has been done for the type of systems the mother standard IEC 61508 (Commission, 1998) is dealing with (see e.g.(Innal, et al., July 2010),(Jin, et al., 2011)). Such a work has not been done yet for automotive safety mechanisms.

In their works, Zhang, Long and Sato(Zhang, et al., 2003) propose models for the representation of multi-channels safety related systems. The Markov models proposed in this paper take into account two kinds of failure: the self-detected ones and the undetected ones. This can be compared to the safety mechanisms diagnostic coverage in this paper. Their models also take into account a “down time” parameter which can be assimilated to the exposure time introduced in ISO 26262 and which is taken into account in our models.

In another article, Yoshimura, Sato and Suyama propose a Markov model to calculate the failure probability of a system without self-diagnostic by taking into account dynamic demand rates (Yoshimura, et al., 2004). Holub and Börcsök enhanced this model by adding the support of the self-diagnostic allowing to distinguish the dangerous detected failures from the undetected ones (Holub & Börcsök, 2009).

In their study, Winkovich and Eckardt propose Markov models to evaluate the failure probability of the IEC 61508 related systems. The models proposed in this paper take into account block equipped with self-test

mechanism, each of them characterized by a self-test period and a diagnostic coverage percentage. However, unlike our models, the proposed models do not take into account the possibility of self-test mechanisms failures (Winkovich & Eckardt, 2005).

3.5 Conclusion

In this chapter, we proposed Markov chains that model the behavior of a large class of automotive Electric and Electronic systems protected by first and possibly second order safety mechanisms. These Markov chains are generic in the sense that the analyst has just to set up the values of parameters such as failure rates and diagnostic coverage to assess a particular system. We report experiments we made to determine the most influential of these parameters.

These Markov chains can serve as reference models for the systems the ISO 26262 standards deal with. Together with our findings on the relative influence of the different parameters, they make it possible to propose approximate models, such as Fault Trees patterns or ad-hoc formulas.

The determination of Fault Tree patterns is of a special interest for most of the analysts which are familiar with this technology, as they allows a convenient manipulation and representation of the various event that can lead to a failure.

In the next chapter, we focus on the presentation and the study of fault tree patterns that allows the approximation of the failure probability that can be calculated with our Markov models.

CHAPTER 4
MAKING IT PRACTICAL

Chapter 4 Making it Practical : Fault Trees Approximations

The calculation of failure probability of the automotive systems and components are mainly performed during the fault tree analyses, so, it is necessary to have good representations of the different elements and data that must be considered. In this chapter, we present and evaluate fault tree patterns that could allow good failures probabilities calculations.

Indeed, the ISO 26262 probabilistic metric also called PMHF, is used for the evaluation of the average failure probability per hour of a system on its functional lifetime, which correspond to the PFH metric defined in IEC 61508 (Commission, 1998). Thus, it mainly relies on the evaluation of the assessed system failure probability F(t), as the PMHF can be approached by F(T)/T where T is the functional lifetime of the assessed system (Innal, et al., July 2010).

This chapter is organized as follows: First, we introduce the fault tree patterns which model our safety mechanisms in Section 5.1. Then, we propose some experiments in order to evaluate the accuracy of each of these representations in Section 5.2 by comparing them with the Markov models results introduced in the previous chapter.

4.1 Fault Tree Patterns Presentation

In this section we present 4 possible fault tree models for representation of the failure of a block and its two safety mechanism. Each of these models features a different way of implementing the second order safety mechanism. The models represented in this section are all implementable using the OpenPSA format (Epstein & Rauzy, 2008).

As the representation of a function failure with a first order safety mechanism can be intuitively obtained and is documented in ISO 26262 part 10 Figure B.4 (ISO 26262, 2011), the main difference between each of these models is in the representation of the second order safety mechanism.

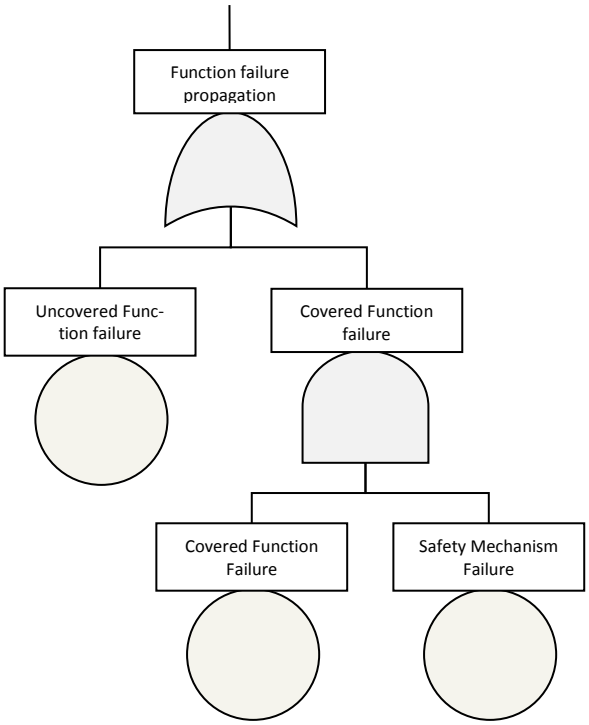

Figure 4:1 ISO 26262 fault tree representation of a function failure with first order SM

4.1.1 FT Model with Classic SM Representation for SM2

In this Model, we consider and represent the second order safety mechanism SM2 as if it was a first order safety mechanism applied on SM1. So, we represent each of them using the classic OR/AND pattern (presented in ISO26262 part 10 (ISO 26262, 2011)).

We consider five basic events:

- The covered part of the block failure, following an exponential law with the parameter DC1 * λ_{HB}
- The uncovered part of the block failure, following an exponential law with the parameter [(1-DC1) * λ_{HB}]
- The covered part of first order safety mechanism failure, following an exponential law with the parameter DC2 * λ_{SM1}
- The uncovered part of the first order safety mechanism failure, following an exponential law with the parameter [(1-DC2) * λ_{SM1}]
- The second order safety mechanism failure, following an exponential law with the parameter λ_{SM2}

It shall be noted that this model (Figure 4:2) can't consider the test interval and the time before going to maintenance. However, as long as the SM2 don't fail, the failure of the covered part of SM1 cannot be propagated.

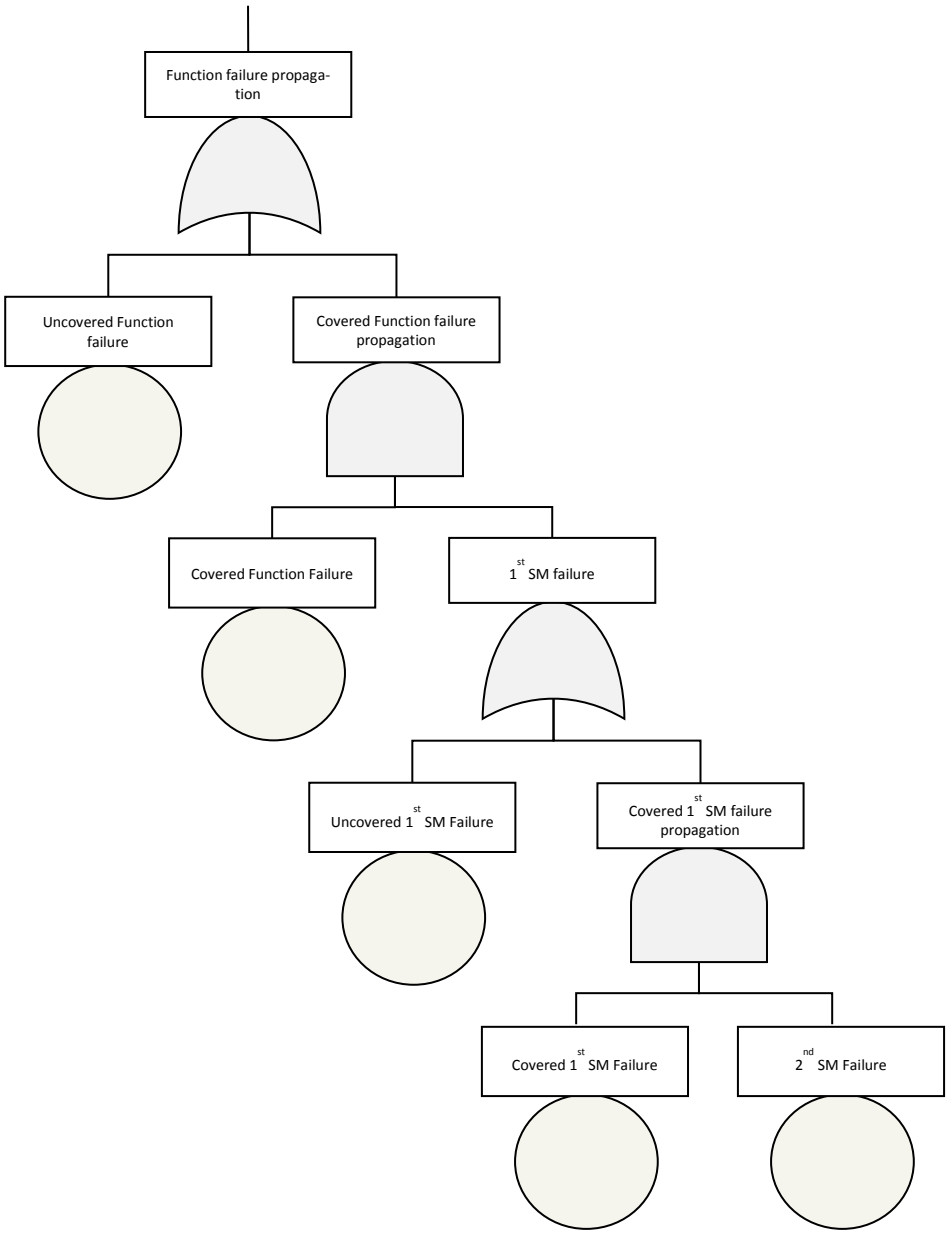

Figure 4:2 Fault tree pattern with a second mechanism represent as a classic safety mechanism

4.1.2 FT Model with Maintenance

In this model, we consider that when the second order safety mechanism SM2 detects the first order safety mechanism failure, it leads to maintenance (and the repair) with a periodic maintenance rate. Thus, this model does not directly consider the possibility of the second mechanism failure but rather focuses on its action.

This fault tree model is based on the generalization and the adaptation of examples from ISO 26262 part 10. Amongst others, the figures B.15 and B.18 are two examples of the use of this model.

However, as a test does not guarantee the maintenance and the repair of the vehicle, we adapted this model to consider the use of maintenance frequencies instead of tests frequencies as shown in those examples.

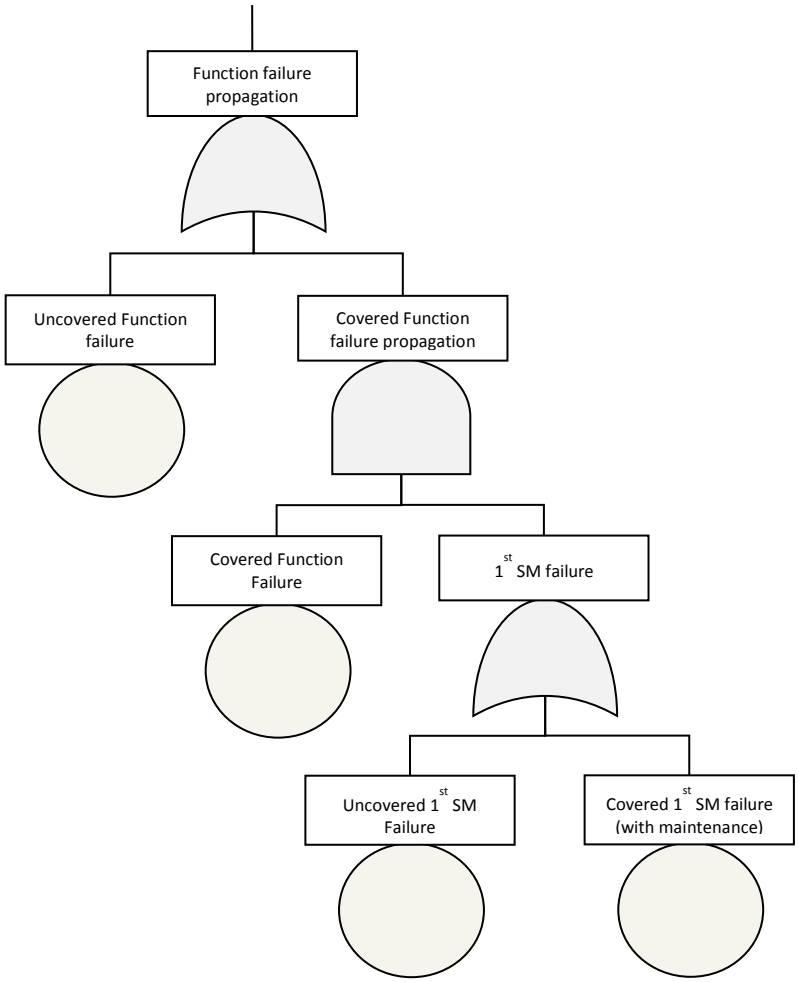

Figure 4:3 Fault tree pattern that takes into account the maintenance action of the 2nd order Safety mechanism

So, we consider four basic events:

- The covered part of the block failure, following an exponential law with the parameter DC1 * λ_{HB}
- The uncovered part of the block failure, following an exponential law with the parameter [(1-DC1) * λ_{HB}]
- The covered part of first order safety mechanism failure, following an exponential law with the parameter DC2 * λ_{SM1}
- The uncovered part of the first order safety mechanism failure, following a GLM distribution with failure rate [(1-DC2) * λ_{SM1}], a reparation rate μ_V and an failure on demand probability (fixed to 0)

This model can be seen as a generalization of the second safety mechanism representation in ISO 26262 part 10 Fault tree examples.

4.1.3 FT Model with Periodic Tests

Like the previous model, this one focuses on the representation of the second order safety mechanism (SM2) action. In this model, in addition to maintenances interval, we also consider SM2 tests periodicity. This is made possible by using the periodic test law defined in OpenPSA (Epstein & Rauzy, 2008).

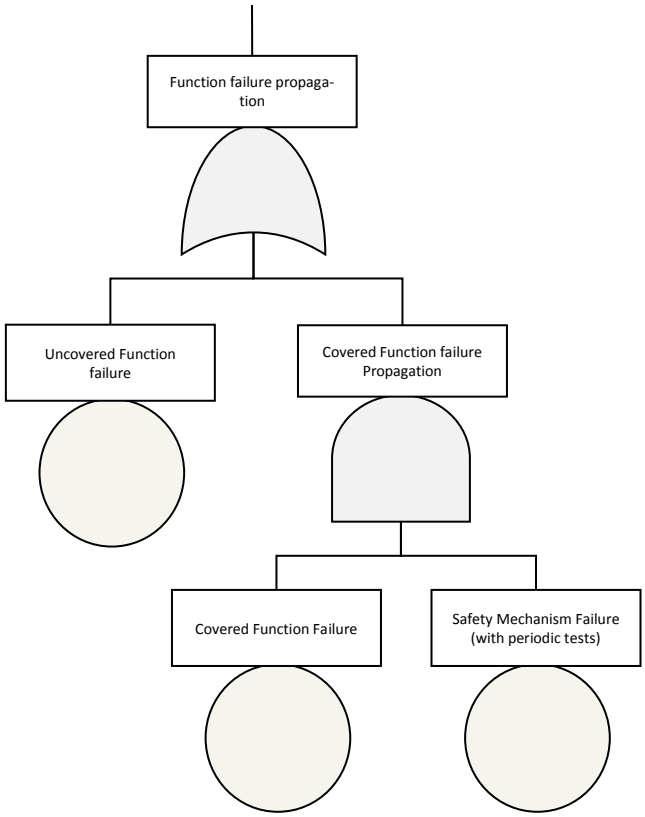

Figure 4:4 Fault tree pattern for the representation of the second order safety mechanism periodical testing behavior

So, we consider three basic events:

- The covered part of the block failure, following an exponential law with the parameter DC1 * λ_{HB}
- The uncovered part of the block failure, following an exponential law with the parameter [(1-DC1) * λ_{HB}]
- The first order safety mechanism failure, following a periodic test law with the following parameters :
    - The failure rate of SM1 when working: λ_{SM1}
    - The failure rate of SM1 when being tested: λ_{SM1}
    - The repair rate of SM1 (when detected): μ_V
    - The Delay between two consecutive tests : $T_V = 1/\delta_V$
    - The Delay before the first test: $T_V = 1/\delta_V$
    - The probability of failure due to the test: 0 (not considered)
    - The duration of the tests : 0 (as the tests are not done during the vehicle service)
    - The availability of the component during the test : 1
    - The probability that detects a failure (if any) : DC2
    - The probability that the component is badly restarted after repair : 0 (as the maintenance is out of the scope of ISO26262)

4.1.4 FT Model without SM2

In this Model, we only represent the block and its first order safety mechanism using the classic OR/AND pattern (ISO 26262, 2011) (as presented in Figure 4:1). We completely ignore the existence of the second order safety mechanism. The purpose of this FT model is to see if the other presented models have a good impact on the failure probability.

So, we consider three basic events:

- The covered part of the block failure, following an exponential law with the parameter DC1 * λ_{HB}
- The uncovered part of the block failure, following an exponential law with the parameter [(1-DC1) * λ_{HB}]
- The first order safety mechanism failure, following an exponential law with the parameter λ_{SM1}

4.2 Experimental Study

In this section, we present the results of our experimentation on the previously presented fault tree patterns in order to test their accuracy, by comparing the obtained failure probability with the failure probability computed with the help of the previously presented Markov diagram.

4.2.1 Realistic Values and Test Sample Description

The values used for the tests are the same than the ones used in the previous chapter for the experiments on the Markov models:

- The lifetime of a vehicle is about 10000 driving hours. We performed most of the calculations for this value.
- The failure rate of hardware blocks (λ_{HB}) stands typically between 10^{-6} and 10^{-8} failures per hour. The failure rates of first and second order safety mechanism (λ_{SM1} and λ_{SM2}) stand typically between 10^{-6} and 10^{-8} failures per hour. We made most of the experiments around these values.
- The diagnostic coverages of first and second order safety mechanisms (DC1 and DC2) are usually rather high (above 90%) but for the purpose of this study we made the DC2 vary significantly. However, we do not consider low DC1 in this experiments, as in these cases, we cannot see at all the influence of the second order safety mechanism.
- The mean journey time ($T_V = 1/\delta_V$) is of course more difficult to estimate. It is considered to be about 1 hour (ISO 26262 part 5, section 9.4.2.3, note2)(ISO 26262, 2011) .
- Similarly, the mean time before the vehicle is taken to the garage when a warning is raised ($T_V = 1/\mu_V$) depends dramatically on the driver. For the purpose of the study, we made it vary significantly from 1 hour (i.e. the journey mean time) to the lifetime of the vehicle.

Considering these characteristics, we built for each FT model a test set of about 1500 samples by the combination of the following values:

Parameter	Values
λ_{HB}	1e-6, 1e-7, 1e-8
λ_{SM1}	1e-6, 1e-7, 1e-8
λ_{SM2}	1e-6, 1e-8
DC1	90%, 95%, 97%, 99%, 100%
DC2	0%, 60%, 95%, 100%
μ_V	0, 0.001, 0.1, 1
δ_V	1

4.2.2 Experimentation Results

By studying these test sets, and comparing their resultant failure probability @10000H (Obtained by using XFTA) with the failure probabilities obtained with the Markov model, we managed to determine the strength and weakness of each FT model.

4.2.2.1 FT Model with Classic SM Representation for SM2

Considering the previously defined samples, we can see that:

The tests on this model show that 425 of the 1440 resulting failure probabilities are more optimistic than the failure probability obtained using the Markov Model.

- This is mainly due to the fact that with this implementation:
  - As long as the SM2 is operational, the failure of the covered part of SM1 cannot be propagated enhancing artificially the failure probability of the component.
  - In opposition to this, as we consider a good δ_v, when take into account that μ_v is null or very low, the SM2 has no influence on the component failure probability.
- However, it should be taken into account that 257 of the samples give optimistic results with a maximum gape of 10%. These samples put apart, 105 of the remaining samples give optimistic probabilities which are less than 10 times lower the ones obtained using the Markov model, giving results that still remain within its range.
- The remaining 63 cases represent the samples with highly optimistic failure probability (at least 10 times lower).
  - All of them have in common a perfect DC1, a really good DC2, and low μ_v.

The tests on this model also show that the remaining 1015 of the resulting failure probabilities are more pessimistic than the ones obtained using the Markov Model.

- These represent the samples where μ_v are not low (higher than 0.001).
- There are 787 of the samples that offer a pessimistic probability with less than 10% difference comparing to the Markov model.
  - These samples are the one with high values for μ_v and an imperfect DC1.
- Also, 147 of the samples give results that are less than 4 times more pessimistic than the ones obtained with the Markov model, giving results within the same scale range.
  - These are the samples with really good DC1 and at least correct μ_v (0.1, 1).
- The remaining 81 of the samples represent the samples with highly pessimistic failure probabilities (At least 4 times more pessimistic).

- All these cases have in common a perfect DC1, correct μ_v.
- The failure probability @10000h obtained with the Markov model for these cases are really low (in the order of 1e-7), so, most of time, even if the FT approximation is highly pessimistic, they should not really impact on an entire system failure.

To conclude, the previous analysis on this fault model shows that it gives good approximations of the Markov failure probability in the following cases:

- Either the diagnostic coverage of the first order safety mechanism is imperfect (DC1<100%).
- Or the values of μ_v are at least correct (0.1, 1).

	Optimistic probabilities in FT				
Difference	[0%, 5%]	[5%,10%]	[10%, 50%]	Less than 10 times	More than 10 times
Samples %	15.21%	2.64%	4.17%	3.13%	4.38%
Samples #	219	38	60	45	63

	Pessimistic probabilities in FT			
Difference	[0%, 5%]	[5%,10%]	Less than 4 times	More than 4 times
Samples %	51.32%	3.33%	10.21%	5.63%
Samples #	739	48	147	81

4.2.2.2 FT Model with Maintenance

Considering the previously defined samples, we can see that the tests on this model present that 289 of the resulting failure probabilities are lower than the ones obtained using the Markov Model offering optimistic results.

- It should be taken into account that 217 of the 1440 samples give optimistic results with less than 10% difference with de Markov model calculation.
- These ones put apart, 63 of the samples give failure probability that are less than 10 times lower than the ones obtained with the Markov model, giving results within its scale range.
  - These are the samples with a maintenance rate μ_v not null, a perfect DC1, and a high DC2 (95%, 100%).
  - This is mainly due to the test intervals which are not taken into account and considered as instantaneous in this FT Model in contrast to the Markov Model.
- The remaining 9 samples represent the samples with highly optimistic failure probabilities (failure probability at least 10 times lower than the ones obtained with the Markov model).
  - These are the samples with a good μ_v (1), perfect DC1 and DC2 (100%).

The tests on this model also show that the remaining 1150 of the samples give more pessimistic results than the Markov model.

- There are 940 of the samples that give pessimistic failure probabilities that have less than 10% difference with the Markov model.
- There are 138 of the samples failure probabilities that are less than 4 times more pessimistic than the Markov model giving results within the same scale range.
  - These are the samples with μ_v not null, good DC1 (97%, 99%, 100%) and low DC2 (0%, 60%).
- The remaining 72 of the samples are the cases where the FT model gives highly pessimistic values.

- All these cases have in common a perfect DC1, average DC2 (60%, 95%) and good μ_v (0.1, 1).
- The failure probability @10000h obtained with the Markov model for these cases are really low (in the order of 1e-7), so, most of time, even if the FT approximation is highly pessimistic, they should not really impact on an entire system failure.

To conclude, the previous analysis on this fault model shows that it gives good approximations of the Markov failure probability in the following cases:

- Either when the maintenance rate and the second order safety mechanism are good.
- Or when the diagnostic coverage of the first order safety mechanism is not perfect.

	Optimistic probabilities in FT				
Difference	[0%, 5%]	[5%,10%]	[10%, 50%]	Less than 10 times	More than 10 times
Samples %	14.65%	0.42%	2.50%	1.88%	0.63%
Samples #	211	6	36	27	9

	Pessimistic probabilities in FT			
Difference	[0%, 5%]	[5%,10%]	Less than 4 times	More than 4 times
Samples %	61.94%	3.33%	9.58%	5%
Samples #	892	48	138	72

4.2.2.3 FT Model with Periodic Tests

First of all, as we consider in this model regular test and maintenance intervals, we tried to determine which value should be the most representative for the maximum failure probability over 10000 hour of service.

This is why we observed the progression of the failure probability in the last hour of mission (between 9999h and 10000h).

Figure 8 represents one of the worst observed cases in our sample in term of fluctuation (case 1) and one of the common cases (case 2), and as we can see in either cases, the value @10000h is a good candidate and the one retained.

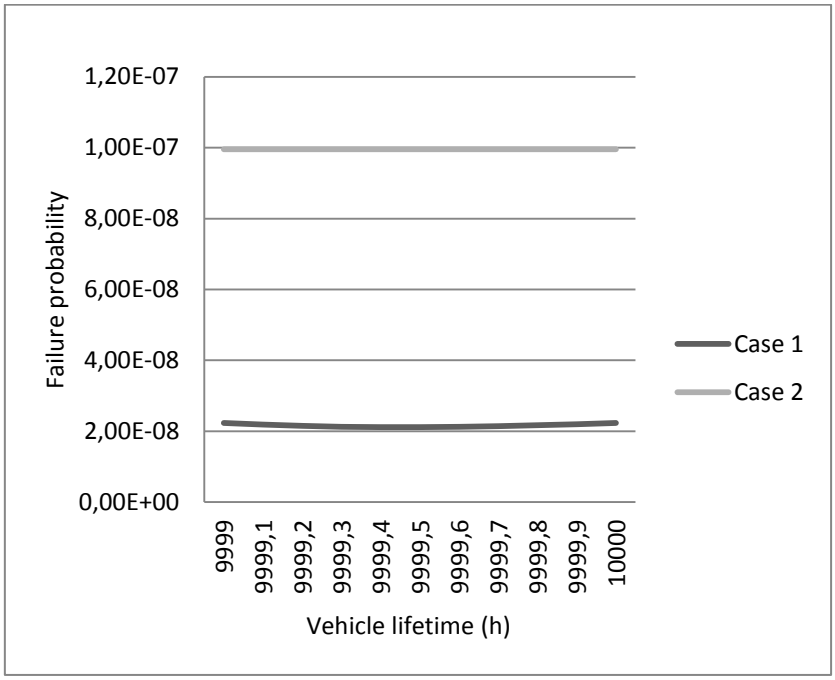

Figure 4:5 Failure probability progression in the last hour of a vehicle lifetime computed with a periodic fault tree model

Considering the previously defined samples, we can see that:

The tests on this model present that 675 of the 1440 resulting failure probabilities are more optimistic than the ones obtained using the Markov Model.

- It should be taken into account that 513 of the samples give optimistic failure probabilities which are less than 10% difference with de Markov model calculation.
- These samples put apart, remaining 144 of the samples give optimistic probabilities which are less than 10 times lower the ones obtained using the Markov model, giving results that still remain within its range.
- Only 18 of the samples are cases where the periodic FT model gives highly optimistic failure probabilities (at least 10 times lower).
    - All these cases have in common a perfect DC1 (100%), average DC2 (60%, 95%) and high μ_v (1).
    - The failure probability @10000 obtained with the Markov model for these cases are really low (in the order of 1e-7), so, most of time, even if the FT approximation is highly pessimistic, they should not really impact on an entire system failure.

The tests on this model also show that the remaining 765 of the resulting failure probabilities are more pessimistic than the ones obtained using the Markov Model.

- There are 653 of the samples that give failure probabilities that are pessimistic with less than 10%.
- The remaining 112 are the samples pessimistic failure probabilities with less than times difference with the results obtained by using the Markov model, giving results within the same scale range (Less than 4 times higher).
    - All these samples have in common a high DC1 (97%, 99%, 100%) a bad DC2 (0%) and a μ_v not null.
    - In fact, the worst cases, the assessed samples with this model were giving results that were only about twice higher.

As we can see from these results, this model offers rather good performances in extreme cases. Indeed, as long as the diagnostic coverage of the first order safety mechanism is imperfect (not 100%), this model offers a really good accuracy in most of cases, and with about 30% divergence in the worst case (36 of 1440 tested cases). However, we didn't manage to extract clear rules that allow the distinction between the cases where it works perfectly and the others.

	Optimistic probabilities in FT				
Difference	[0%, 5%]	[5%,10%]	[10%, 50%]	Less than 10 times	More than 10 times
Samples %	34.38%	1.25%	6.46%	3.54%	1.25%
Samples #	495	18	93	51	18

	Pessimistic probabilities in FT			
Difference	[0%, 5%]	[5%,10%]	Less than 4 times	More than 4 times
Samples %	45.35%	1.53%	6.25%	0%
Samples #	653	22	90	0

4.2.2.4 FT Model without SM2

Considering the previously defined samples, we can see that the tests on this model present that 42 of the 1440 resulting failure probabilities are more optimistic than the ones obtained using the Markov Model.

- Each of these samples has an enhanced lower failure probability with less than 2% of difference.
    - All these samples have in common a μ_v which is null, a perfect DC1 and a DC2 which is not null.

The tests on this model also show that the remaining 1398 of the samples obtained with this model give pessimistic failure probabilities.

- There are 854 of the samples that give failure probabilities which are pessimistic with less than 10% difference with the Markov model.
    - Each of them has in common either an imperfect DC1 or a perfect DC1 with a μ_v that is null.
- Also, 382 of the samples give results that are less than 4 times more pessimistic than the ones obtained with the Markov model, giving results within the same scale range.
    - Each of them has in common a μ_v that is not null.
- The remaining 162 of the samples represent the samples with highly pessimistic failure probabilities (At least 4 times more pessimistic).
    - All these samples have in common a perfect DC1, with DC2 and μ_v not null.

As we can see from these results, this model is really pessimistic. This is normal as we do not consider the second order safety mechanism at all, which perfectly correspond to the cases where our δ_v and μ_v are low, as the second mechanism does not impact strongly on our failure probability.

	Optimistic probabilities in FT				
Difference	[0%, 5%]	[5%,10%]	[10%, 50%]	Less than 10 times	More than 10 times
Samples %	2.92%	0.00%	0.00%	0.00%	0.00%
Samples #	42	0	0	0	0

	Pessimistic probabilities in FT			
Difference	[0%, 5%]	[5%,10%]	Less than 4 times	More than 4 times
Samples %	59.31%	8.19%	18.33%	11.25%
Samples #	854	118	264	162

4.2.3 Synthesis

Given the previous results, we can see that there is no model allowing the approximation in all cases, however, we can provides some recommendations on which model fits the best for certain cases to obtain a good approximation of the Markov failure probability:

- Each of these model show good result when the diagnostic coverage of the first order safety mechanism is not high (DC1<90%)
- The model with Classic OR/AND pattern for the representation of SM2 failure shows reasonably good result when DC2 is high and μ_v is good.
- The model with a GLM law taking into account the maintenance rates shows reasonably good results when it comes to the approximation of the failure probability when μ_v is not null and DC2 is.
- The model with periodic test law shows the best average accuracy, however, there are no rules or explicit conditions allowing us to separate the cases where this model is not accurate from the others, so, its applicability really depends on expert judgment. It should be noted however that as long as the diagnostic coverage of the first order safety mechanism is imperfect (not 100%), this model offers a really good accuracy in most of cases, and with about 30% divergence in the worst case (36 of 1440 tested cases).
- The model with no SM2 representation is the model which gives the most pessimistic approximations of the exact failure probability, and shows particularly good results when μ_v is null. However using it in the other cases could highly degrade the estimated failure probability in comparison to the exact one.

This is why we recommend the use of each model only when the above conditions are met. When none of these conditions are met, the periodic test law model is the one that should show the better approximation in most of cases.

4.3 Conclusion

In this chapter we presented fault tree patterns for the representation of a large class of automotive electric and electronic function with its safety mechanism (first and second order). By comparing them to markov chains which can serve as benchmark models for ISO 26262, we tried to identify their strengths and weaknesses. This led us to the conclusion that each of these provides good approximations only in some specific cases.

The fault tree patterns presented here will serve in the next chapter to present our methodology that allows us to compute ISO 26262 specific metrics and our process for the FMEDA generation from fault tree like patterns.

CHAPTER 5

SPECIFICS DEVELOPMENTS FOR ISO26262 SAFETY ANALYSES

Chapter 5 Specific Developments for ISO26262 Safety Analyses

In the previous chapter, we presented different fault tree patterns. These patterns make it possible to represent the failure behaviour of automotive safety mechanisms and provide a good accuracy in most of the cases. In this chapter we will present some specific developments made in the scope of ISO26262 standard deployment.

5.1 Overall Process

The objective we had was to perform all the quantitative assessments by means of fault trees.

We first introduce a custom coverage gate for fault trees. This gate allows the generation of each of the previous patterns.

Using this coverage gate, we present a method that allows ISO26262 architectural metrics calculation. Then, we present our work that makes quantitative FMEDA generation possible.

To finish we present our coherence check methodology. This check makes it possible to generate a complete FMEDA (quantitative and qualitative) and provides assistance in the safety analyses process.

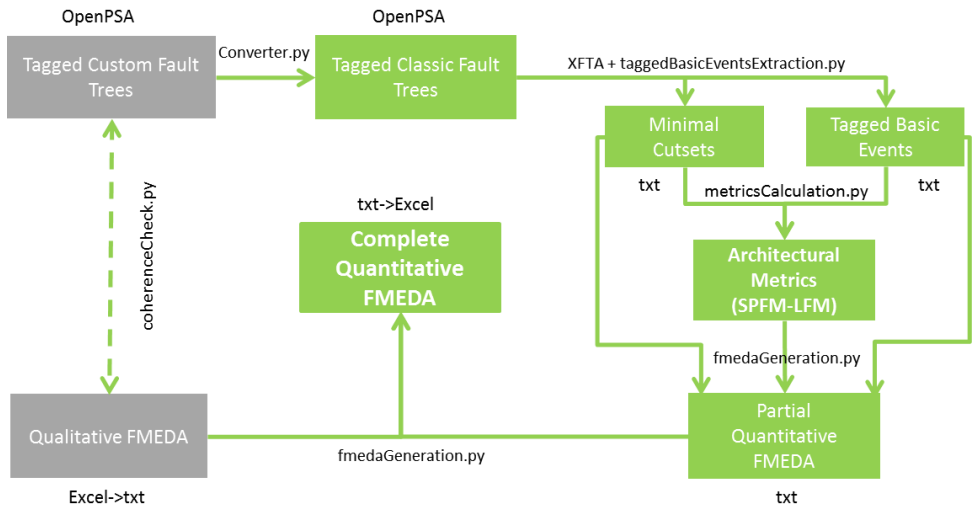

Figure 5:1 ISO26262 Specific developments plan for safety analyses generation

Figure 5:1 gives an overview of the development plan that we followed for the safety analyses generation. The inputs are in OpenPSA format and formatted txt. The processing are done using python scripts and XFTA (for the cut-sets extraction) (Rauzy, 2012).

5.2 Coverage Gate

To ensure the compatibility of our works with each of the previously introduced patterns, we first built fault tree like patterns. These patterns use a custom gate defined here Coverage Gate. Its use is to represent the coverage relation of a safety mechanism.

The idea behind such kind of representations is not new, indeed, similar works have already been realized on binary decisions diagrams (BDD) (Myers & Rauzy, 2008) (Amari, et al., 2008).

We define this coverage gate as an asymmetric gate that take three inputs:

- the first input is always the covered element,
- the second input is always a safety mechanism,
- the third one is a parameter labelled as “DC” and contains the value of the diagnostic coverage of the safety mechanism toward the basic event.

This Coverage Gate and its usage have been designed to be “object orientated”, as each attribute is placed in the element which it corresponds to. For example, the diagnostic coverage (DC) - which depends of both the safety mechanism and the elements it covers - is placed on the Coverage Gate, allowing an easy access to this data. This was important, as one of the main purposes behind the construction of such a pattern was to be able to generate more classical fault trees from it.

Also, the practicality of such pattern is also reinforced as it is more compact than the classic ones. This makes it more handlable during the safety analyses and less prone to misuses.

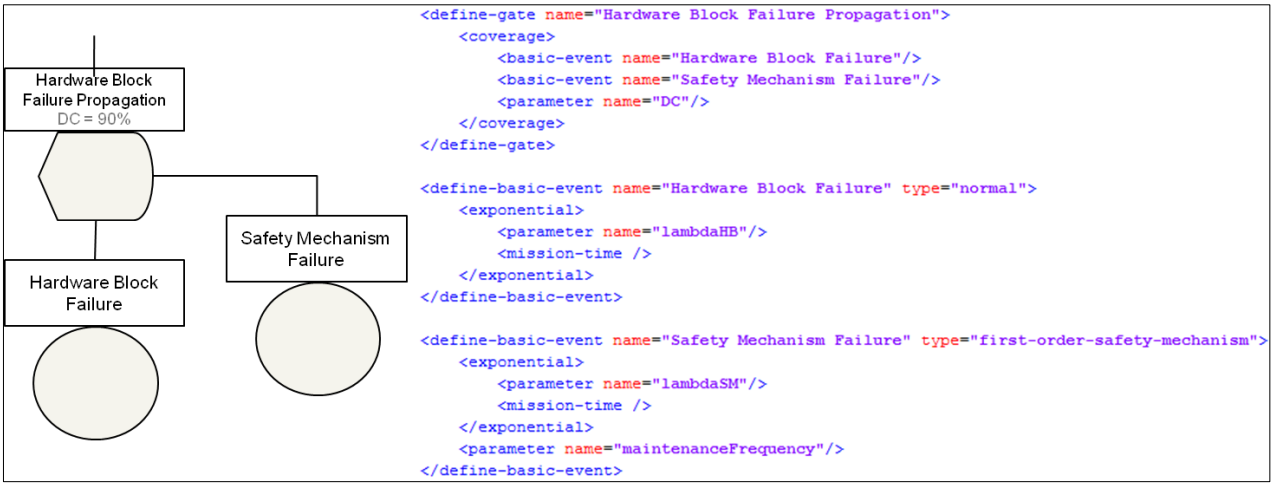

Figure 5:2 Coverage gate custom pattern (Graphical representation + Open-PSA XML)

Indeed, from the XML definition of a fault tree using the Coverage Gate, we can easily obtain each of the previous patterns by using a simple parsing algorithm (as displayed in the figure bellow).

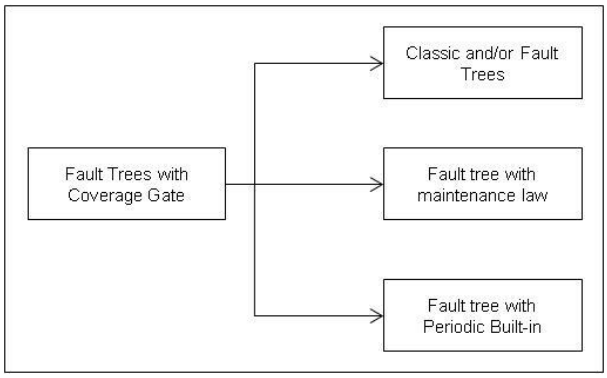

Figure 5:3 Coverage Gate Fault Free existing parsing and translation possibilities

For example, to generate a fault tree that uses the classic or/and pattern from a fault tree using coverage gate, these are the main steps:

- We start browsing a fault tree from its top gate to the bottom, until we find the last accessible coverage gate,
- We duplicate all the branch corresponding to the first element of the Coverage Gate:
    - We add the suffix “(uncovered)” for each element of the first one and weight their failure rates with the inversed diagnostic coverage value (third parameter of the coverage gate),
    - We add the suffix “(covered)” for each element of the second one and weight their failure rates with the diagnostic coverage value,
- We create an And Gate with the following sub elements: the covered branch and the safety mechanism failure (second element of the coverage gate),
- We replace the Coverage Gate with a newly created Or Gate containing the two following sub elements: the uncovered branch and the previously created Or Gate.
- We repeat these operations on the new tree, until there is no Coverage Gate left.

There are some subtleties that are not detailed here, for example, the way that we deal with the And Gates that are - directly or indirectly - under a Coverage Gate. In fact, when generating the covered and uncovered branch, we only ponderate the first sub element of this And Gate, the other elements stay untouched.

From this design choice, two things result: the diagnostic coverage will have a realistic impact on the second order cut sets or above; but, a safety mechanism will never be covered by another safety mechanism which is in a Coverage Gate above it in the tree.

```
<define-gate name="Hardware Block Failure Propagation">
    <or>
        <basic-event name="Hardware Block Failure (Uncovered)"/>
        <basic-event name="Hardware Block Failure Propagation (Covered)"/>
    </or>
</define-gate>

<define-gate name="Hardware Block Failure Propagation (Covered)">
    <and>
        <basic-event name="Hardware Block Failure"/>
        <basic-event name="Safety Mechanism Failure"/>
    </and>
</define-gate>

<define-basic-event name="Hardware Block Failure (Uncovered)">
    <exponential>
        <mul>
            <sub>
                <int value="1"/>
                <parameter name="DiagnosticCoverage3"/>
            </sub>
            <parameter name="lambdaHB"/>
        </mul>
        <mission-time />
    </exponential>
</define-basic-event>
```

```
<define-basic-event name="Hardware Block Failure (Covered)">
    <exponential>
        <mul>
            <parameter name="DC"/>
            <parameter name="lambdaHB"/>
        </mul>
        <mission-time />
    </exponential>
</define-basic-event>

<define-basic-event name="Safety Mechanism Failure">
    <exponential>
        <parameter name="lambdaSM"/>
        <mission-time />
    </exponential>
    <parameter name="maintenanceFrequency"/>
</define-basic-event>
```

Figure 5:4 OpenPSA code obtained when generating Classic Or/and tree from a coverage gate pattern

We present in the Annex A, a bigger size example based on the second safety goal defined in ISO 26262 Part 5 Annex E (ISO 26262, 2011).

In the following section, we present our ISO 26262 specific developments which are based on this pattern. After the introduction of the ISO 26262 architectural metrics, we will present our methodology to compute them from fault trees.

5.3 Architectural metrics calculation

ISO 26262 defines two architectural metrics (Single Point Fault Metric and Latent Fault Metric) to estimate the proportion in a component/system of certain types of fault ~~of~~ causing a certain unwanted event with regard to all the faults that can attain that component.

5.3.1 ISO 26262 Architectural Metrics presentation

In order to present these two metrics, it is necessary to introduce the different fault types that can attain the automotive systems. The table below presents the different types of hardware faults which are considered in ISO26262 standard and their corresponding and their corresponding categories.

Table 5:1 ISO 26262 Part 5 Annex C definition of fault types

Fault	**Description**	**Failure Rate**
Basic faults	Each fault that can attain a hardware part in our systems can be considered as a basic fault either if it causes the system failure or not. It can be considered as the sum of all the failure modes that an assessed hardware part can be subject to	λ
Singles Point Fault	A fault can be considered as a Single Point Fault if its occurrence directly implies the occurrence of an unwanted event in the system	λ_{SPF}
Residual Faults	It is a fault that directly causes an unwanted event, even if the responsible hardware block failure is covered by a safety mechanism	λ_{RF}

	probably due an imperfect diagnostic coverage	
Multiple Point Faults	These are faults that cannot directly lead to the occurrence of the undesired event, but instead, their combination can do it	λ_{MPF}
Multiple Point Fault (Latent)	These are the Multiple Point Faults that cannot be detected nor perceived, so they stay latent in the system until the apparition of other multiple fault that can lead to the undesired event	$\lambda_{MPF,Latent}$
Multiple Point Fault (Perceived or detected)	These are the Multiple Point Faults that can be perceived by the driver, either by the help of a safety mechanism signalization or by performance degradation	$\lambda_{MPF,Perceived\ or\ Detected} = \lambda_{MPF} - \lambda_{MPF,Latent}$
Safe Fault	These are the faults that cannot lead to the Assessed unwanted event. In practice if a Multiple Point fault requires several other ones for the occurrence of an Unwanted Event than we consider it as a Safe fault	$\lambda_S = \lambda - \lambda_{RF} - \lambda_{SPF} - \lambda_{MPF}$

5.3.1.1 Single Point Fault Metric

The Single Point Fault Metric (SPFM) represents the proportion of random hardware faults that do not directly lead to the occurrence of undesired event, thus, it represents the robustness of the item that is assessed to the single point and residual faults. It is defined in the ISO 2626 by the following formula:

$$1 - \frac{\sum_{SR,HW}(\lambda_{SPF} + \lambda_{RF})}{\sum_{SR,HW}\lambda} = \frac{\sum_{SR,HW}(\lambda_{MPF} + \lambda_{S})}{\sum_{SR,HW}\lambda}$$

Figure 5:5 Single Point Fault Metric Formula

As we can see, for the calculation of this metric, we need to be able to identify the following:

- The Single point faults and residual faults failure rates for the numerator calculation ($\lambda_{SPF} + \lambda_{RF}$),
- The sum of all the safety related failure rates (λ) for the denominator.

5.3.1.2 Latent Fault Metric

The Latent Fault Metric (LFM) reflects the robustness of a system with regard to the latent fault that leads to a specific undesired event. It represents the proportion of multiple faults that do not remain unnoticed in the system and that could lead to the occurrence of a hazard. It is defined in the ISO 26262 by the following formula:

$$1 - \frac{\sum_{SR,HW}(\lambda_{MPF,latent})}{\sum_{SR,HW}(\lambda - \lambda_{SPF} - \lambda_{RF})} = \frac{\sum_{SR,HW}(\lambda_{MPF,perceived\ or\ detected} + \lambda_{S})}{\sum_{SR,HW}(\lambda - \lambda_{SPF} - \lambda_{RF})}$$

Figure 5:6 Single Point Fault Metric Formula

As we can see, for the calculation of this metric, we need to consider and be able to identify the following:

- The sum of all Latent Multiple Point Faults failure rates ($\lambda_{MPF,Latent}$) for the numerator calculation,
- The sum of all the Safety Related Faults, the Single Point Faults and the residual faults failure rates ($\lambda - \lambda_{SPF} - \lambda_{RF}$), for the denominator.

5.3.1.3 Architectural Metrics Simplified Calculation Method

In reference (L'Hostis, 2013), L'Hostis proposes a simplified method for the calculation of the ISO 26262 architectural Metrics. This method makes it possible to get approximated values without having to dive down into hardware parts, hence saving time for the analysts. The obtained values are always pessimistic and the method is thus conservative.

In fact, the approximation comes from the fact that L'Hostis considers only what he calls Safety Related Basic Event failure rates. These failure rates correspond to the different failure modes of hardware blocks that lead to the occurrence of basic events, which themselves lead to a failure of the whole system.

The failure rate associated to a hardware block failure corresponds to the aggregation of the failure rate of the hardware parts composing it, and is considered as the same failure rate as the one associated to its corresponding basic event.

Thus, since the failure rates that are contained into the Fault Tree model are exactly the same as those considered in l'Hostis' works, the results of the two methods should be the same.

The benefit of the approach that we present here is that all of the relevant metrics can be calculated from the unique fault tree model and ad-hoc separated study is not anymore necessary. But first, we must present a process to categorize the basic events. This will be the object of the following section.

5.3.2 Architectural Metrics Calculation from fault trees

In this section, we will present our methodology for the computation of the Single Point Fault and Latent Fault Metrics from fault trees. To our knowledge, only few investigations have been done in this direction. We can quote basically only one communication in CTI Conference 2012 (Stanyer, 2012). There are however works that were led in parallel of this thesis in the SAFE Project for the calculation of architectural metrics from models (ITEA2, 2013).

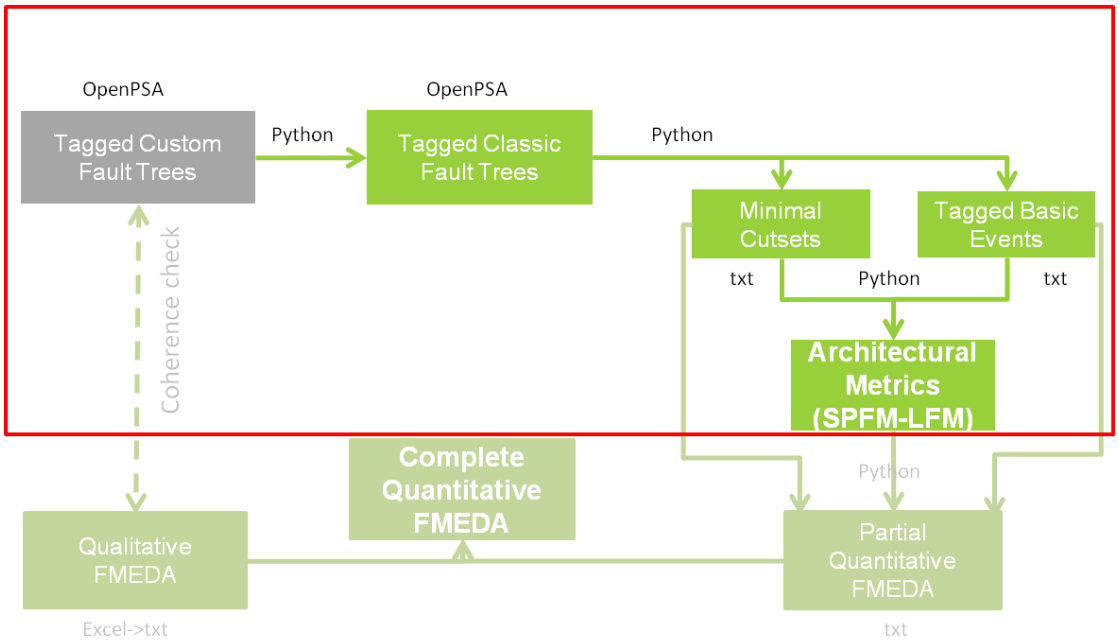

Figure 5:7 ISO 26262 Specific developments plan for architectural metrics generation

In the first subsection we first present our basic events classification method, then we present a method based on tags that allows complementing fault trees data and thus compute the architectural metrics from them.

5.3.2.1 Fault Classification

In order to compute the two architectural metrics from fault trees, it is necessary to distinguish different categories of basic events. The chart presented below, makes it possible for the analyst to split Basic Events into 3 categories:

- Single Point Faults and Residual Faults that show up in the numerator of the SPFM definition.
- Latent Faults that show up in the numerator of the LFM definition.
- And finally, the remaining faults that are safety related, but that show up neither in the SPFM nor the LFM numerators.

This is a practical reinterpretation of the fault diagram in ISO 26262 Part 5 Annex B(ISO 26262, 2011).

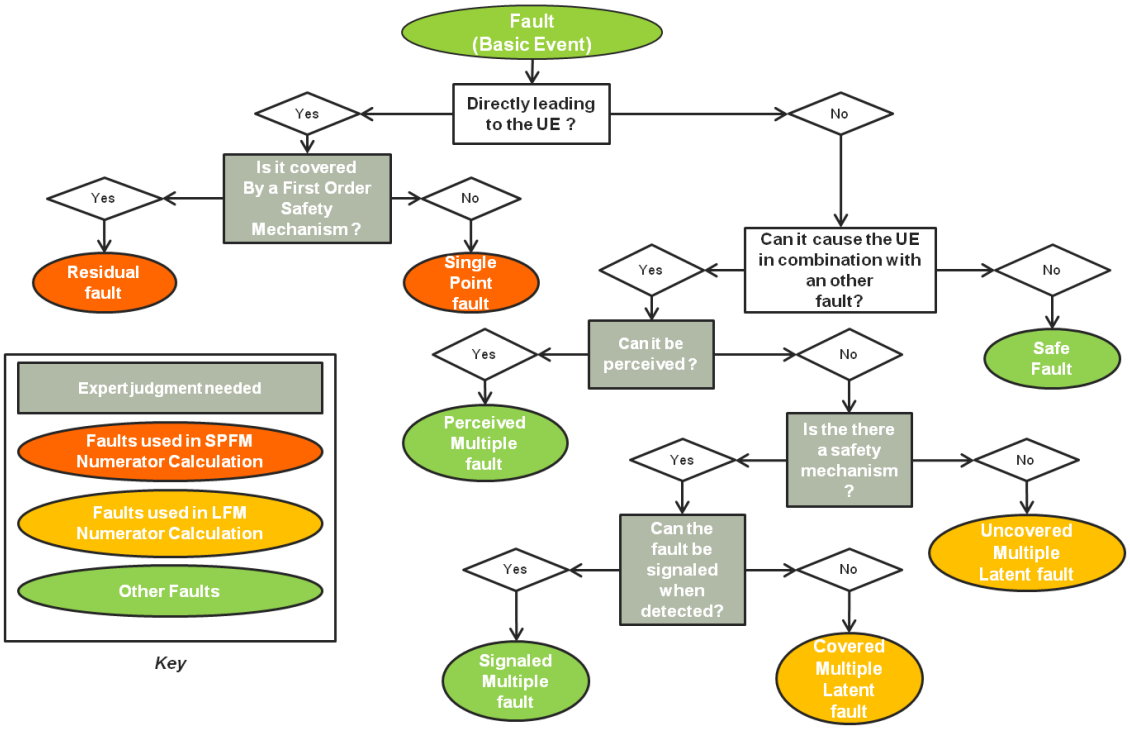

Figure 5:8 Rearranged fault classification diagram

In Figure 5:8 we clearly see that there are 4 important data that allow us to determine the category of fault or Basic Event:

- If the fault directly leads to the component failure, it is taken into account in the SPFM tor $(\lambda_{SPF} + \lambda_{RF})$.
- If it can lead to the component failure in combination with another fault (λ_{MPF}) :
    - If it can be perceived by the driver (for example, performance degradation) it's a Perceived Multiple Point Fault $(\lambda_{MPF,Perceived})$.
    - Else, If the fault can lead to the component failure and is covered and detected by a safety mechanism :
        - If the safety mechanism can alert the driver, than it is considered as a detected fault$(\lambda_{MPF\ ,Detected}\ )$.
        - Else, it is a latent fault $(\lambda_{MPF,Latent})$.
    - If the fault does not lead at all to the assessed unwanted event, it's directly categorized as safe fault (λ_S).

On one hand, as seen in Figure 5:8, the basic events failure rates that are taken into account in the SPFM numerator can be easily extracted from the first order cut sets of a fault tree. On the other hand, there is no way to efficiently identify the latent multiple points from the detected ones using a simple fault tree for the LFM calculation. This is why we define tags based on given fault tree patterns in order to be able to categorize faults and failures latency and thus, computing the second architectural metric.

5.3.2.2 Tag Based Approach Presentation

In this section we present a method based on tagged fault trees for the calculation of the simplified ISO 26262 architectural metrics. Based on the previously defined classification diagram, we first define tags

that allow us to categorize fault latencies, and then we present how to use them on our custom fault trees that use Coverage Gates.

5.3.2.2.1 Tag Definition

Based on the fault classification presented in Figure 5:8, we managed to define three tags:

- The first one is used to indicate if the occurrence of a basic event can naturally be perceived, we associate to it a value between 0 and 100, which represent the percentage of the perception probability.
    - We represent it by the form: [Perceived,X], where X is a percentage between 0 and 100%.
    - When using the OpenPSA XML, we represent this tag by adding the attribute perception-rate=”X” to the basic event definition.
- The second one is used to indicate if the basic event occurrence is covered by a safety mechanism that alerts when it detects its failure. This tag could be binary, however, we chose to associate a percentage to it in order to take into account the probability that the driver misses or ignores the signal/alert.
    - We represent it by the form: [Signaled,X], where X is a percentage between 0 and 100%.
    - When using the OpenPSA XML, we represent this tag by adding the attribute signalisation-rate=”X” to the basic event definition.
- If the basic event occurrence correspond to a 2nd order safety mechanism failure, it is necessary to tag it in order to signify that it is considered as a safe fault. Indeed, as the second order safety mechanism have no direct impact on failure propagations, they are considered as safe faults in ISO 26262.
    - We use the tag [SecondOrder]. This tag can be placed automatically in our custom pattern, as the second order safety mechanisms are the only ones that have a test interval.
    - When using the OpenPSA XML, we represent this tag by adding the attribute safety-mechanism -type= “second-order” to the basic event definition.

5.3.2.2.2 Tag Based Algorithm for Architectural Metrics Computation

Before the presentation of the architectural metrics computation algorithm, we need to introduce which inputs are needed.

The algorithm considers two inputs: the minimal cut sets of the fault tree which we want to assess, and the list of all its basic events with their corresponding tags.

So, the first thing to do is to build a fault tree using Coverage Gates for the representation of the safety mechanisms actions, after that, we add the tags were it is necessary following their definition.

After that, we generate classic and/or fault trees with this tag inheritance policy:

- If an event is tagged with the [Perceived.X] tag in the custom fault tree, and if this event is under a coverage gate, then both the covered and uncovered event generated from it inherit of this tag when converting the Coverage Gate to an Or/And Classic pattern;
- If an event is tagged with the [Signaled.X] tag in the custom fault tree, and if this event is under a coverage gate, then only the covered event generated from it inherit of this tag when converting the Coverage Gate to an Or/And Classic pattern.

Next to this, we extract the tagged basic events list and the minimal cut sets from this newly generated tagged classic fault tree.

And finally we apply the following algorithm:

```
Ignored = {};
Perceived = {};
Single = {};
Signaled = {};
Latent = {};
Safe = {};

Extract all the cut-sets from the fault tree with their tags;
for each cut-set with an order > N
        for each basic event BE in the cut-set
                Ignored = Ignored + {BE};
        end ;
end ;

for each cut-set with an order = 1
        Single := Single + {Be};
end ;

for any other cut-set order
        for each basic event Be in each cut-set
                if the basic event is tagged by [SecondOrder]
                        Safe :=  Safe + {Be};
                elseif the basic event has the tag [Perceived,X] (where X is a number)
                        BE.coefficient := X from [Perceived,X];
                        Perceived := Perceived + {Be};
                elseif the basic event has the tag [Signaled,X] (where X is an number)
                        BE.coefficient := X from [Signaled,X];
                        Signaled := Signaled + {Be};
                else
                        Latent := Latent + {Be} ;
                end ;
        end ;
end ;
LambdaIgnored = Sum of Be.Lambda in Ignored;
LambdaSingle = Sum of Be.Lambda in Single;
LambdaLatent = (Sum of Be.Lambda in Latent) + (Sum of Be.Lambda*(1-Be.coefficient) in Perceived and Signaled);
LambdaSafe = (Sum of Be.Lambda on Safe) + (Sum of Be.Lambda*Be.X on Perceived and Signaled);
```

In this algorithm, we ignore the cut sets that have an order above “N”, this is in line with an ISO 26262 requirement that allows to potentially consider fault as safe fault when the combination order is high enough to be improbable. (most of time, we consider that N = 3).

As we can see, the first thing to do is to extract all the cut set in the fault tree. Then, based on the tags, we calculate the sum of the failure rates corresponding to each category of fault.

By using this algorithm result, we're able to compute the architectural metric parameters assuming the following equivalences:

- $\sum_{SR;HW}(\lambda_{SPF} + \lambda_{RF}) = LambdaSingle$; Because the first order cut-sets represent the fault directly leading to the unwanted event.
- $\sum_{SR;HW}(\lambda_{MPF,Latent}) = LambdaLatent$; Because this represent the part of the basic event which are neither perceived nor signaled.
- $\sum_{SR;HW}(\lambda_{safe}) = LambdaIgnored + LambdaSafe$; Because this represent the combination of basic event that are of a too high order to be considered or that are perceived or signaled.
- $\sum_{SR;HW}(\lambda_{SRBE}) = LambdaIgnored + LambdaSafe + LambdaLatent + LambdaSingle$; The sum of safety related basic event failure rates as defined in reference (L'Hostis, 2013).

Thus, we have:

$$\boldsymbol{SPFM = 1 - \frac{LambdaSingle}{LambdaTotal}} \quad , \quad \boldsymbol{LFM = 1 - \frac{LambdaLatent}{LambdaTotal - LambdaSingle}}$$

5.3.3 Application Example

In this sub-section, we will present an example of the usage of the previously presented tag based approach. We will first build an example fault tree using "coverage gates" based on the Vehicle management unit presented in Figure 3:1: We consider that the electric motor receives a wrong three phase current if either the Electric Motor inverter or the Vehicle Management Unit is attained by a failure.

For the purpose of simplification, we will assume that the Electric Motor Inverter failure is a basic event. We then obtain the fault tree presented in Figure 5:9.

Also, we consider the PTU and the TCU as independent, because their two functions are never executed at the same time even if they are supported by the same hardware (as seen in Figure 3:1).

The chosen numerical values in this example are not realistic but their range is; the purpose behind this is to provide an easily computable and understandable example where we could see the influence of each parameter.

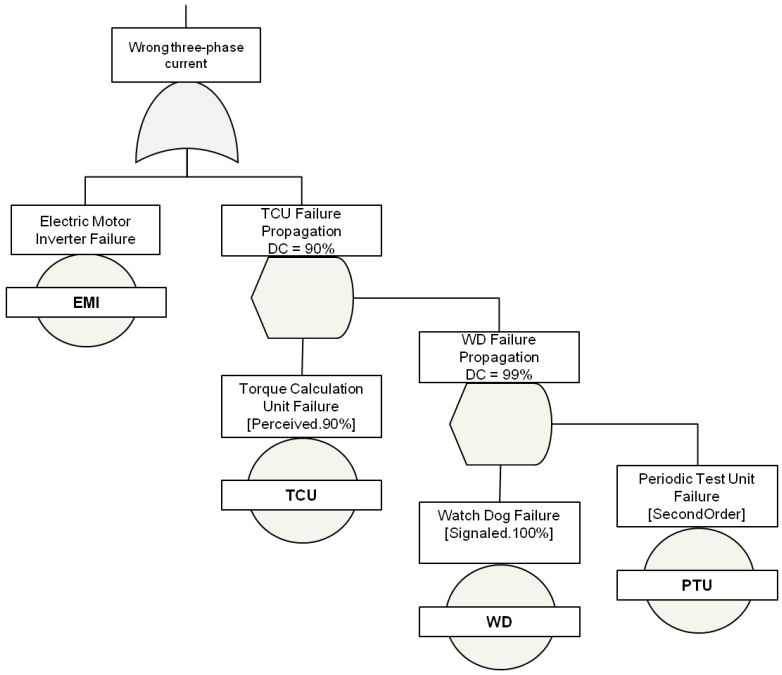

Figure 5:9 Tagged fault tree simplified example for the representation of the generation of a wrong three-phase current for an electric motor

The basic events presented in this fault tree (Figure 5:9) are all characterized by their failure rates (Table 5:2). The coverage of each safety mechanism toward the basic event it covers is presented on the corresponding coverage gate (90% for the Watch Dog and 99% for the Periodic Test Unit).

Table 5:2 Basic events failure rates for the wrong three phase current generation fault tree

Basic event	**Failure rate name**	**Failure rate value**
Electric Inverter Unit failure	λ_{EIU}	$5e^{-8}$
Torque Calculation Unit failure	λ_{TCU}	$1e^{-6}$
Watch Dog failure	λ_{WD}	$1e^{-7}$
Periodic Test	λ_{PTU}	$5e^{-8}$

As the Periodic Test Unit is a second order safety mechanism, it should also be characterized by a test frequency and a maintenance frequency.

We then proceed to the minimal cut sets extraction as presented in Section 5.2.2. We obtain four cut sets composed with 5 basic events. Two first order cut sets:

- The first one is composed by the Electric Inverter Unit failure with a failure rate λ_{EIU},
- The second one is composed of the uncovered part of Torque calculation unit failure. Its failure rate is obtained by inverting the related DC: $\lambda_{UTCU} = 10\% \lambda_{TCU}$.

We also obtain a second order cut set composed by:

- The covered part of the Torque calculation Unit, with a failure rate $\lambda_{UTCU} = 90\% \lambda_{TCU}$.
- The uncovered part of the Watch Dog failure, with a failure rate $\lambda_{UWD} = 1\% \lambda_{WD}$.

There is also a third order cut set composed by:

- The covered part of the Torque calculation Unit, with a failure rate $\lambda_{UTCU} = 90\% \lambda_{TCU}$,
- The covered part of the Watch Dog failure, with a failure rate $\lambda_{DWD} = 99\% \lambda_{WD}$,
- The Second order safety mechanism failure $\lambda_{PTU.}$

For each of the basic events that we obtain in this cut sets, we inherit the tags of their original event. Then we apply the previously presented algorithm. The obtained results are the following:

LambdaIgnored	=	0
LambdaSingle	=	$\lambda_{UTCU} + \lambda_{EIU} = 5e^{-8} + 1e^{-7} = 1.5e^{-7}$
LambdaLatent	=	$\lambda_{UWD} + (1\text{-}90\%)\lambda_{DTCU} + (1\text{-}100\%)\ \lambda_{DWD} = 1e^{-9} + 10\%\ 9e^{-7} + 0 = 9.1e^{-8}$
LambdaSafe	=	$\lambda_{PTU} + 90\%\ \lambda_{DTCU} + 100\%\ \lambda_{DWD} = 5e^{-8} + 90\%\ 9e^{-7} + 9.9e^{-8} = 9.59e^{-7}$

Thus, we finally obtain:

$$\boldsymbol{SPFM = 1 - \frac{LambdaSingle}{LambdaTotal} = 87.5\%} \quad , \quad \boldsymbol{LFM = 1 - \frac{LambdaLatent}{LambdaTotal - LambdaSingle} = 88,75\%}$$

Other tests have been led on more detailed examples directly extracted from the ISO 26262; the obtained results were exactly the same as the ones obtained with approach proposed by L'Hostis. Although, these are not the exact calculation of the metrics, it offers good pessimistic approximations. In Annex A, we present a more in depth example based on the second safety goal defined in ISO 26262, Part 5, Annex E.

5.4 FMEDA Generation Methodology

The Failure Mode, Effects and Diagnostic Analyses (FMEDA) is a systematic technique for failure analyses, it is composed of two separate analyses:

- The qualitative side of a FMEDA corresponds to Failure Modes and Effects Analyses (FMEA). Its principle is to identify all the critical failure modes of a hardware blocks (the Basic Events) and the way they propagate to cause the component Undesired Events – which violate a safety goal – in order to define adequate safety mechanisms at component level.
- The Quantitative side of a FMEDA verifies the quantitative requirements allocated to a component for a particular component UE. In the Valeo implementation of the ISO26262 process, one of the main issues that are addressed by the quantitative FMEDA, is the computation of the ISO26262 architectural metrics.

In the Valeo process, the FMEA is generally the first analysis done after the PHA, as it allows to approach the exhaustiveness for the identification of the failure modes that could lead to a system dangerous state. So, instead of trying to generate it from our fault tree analysis (with tags), we found it more interesting and useful to be able to check the coherence of the data found in those two views and to generate a report addressing this subject allowing to enhance their exhaustivity.

As opposed to that, as the tagged fault trees contain all the data for the Metrics Generation, we also developed a methodology for quantitative FMEDA generation from those fault trees.

5.4.1 Qualitative & Quantitative FMEDA Templates

In this section we will present the template of qualitative and quantitative FMEDA that are used in our generation process.

The following table represents the qualitative FMEDA header. Its role is to elicit all the failure modes that can attain the blocs composing a system, and tell if they can lead to the violation of a safety goal. This is the description of the different fields composing it:

- **(A1):** block name corresponds to the name of the considered component or function,
- **(A2):** description of the Function realized by the considered block. There may be more than one function per block. Usually, a function is linked to an output of the block,
- **(A3):** potential failure mode of the functional block,
- **(A5):** failure rate associated to the failure mode,
- **(A6):** contains the name of an undesired event If the failure mode leads to it

Table 5:3 Qualitative FMEDA header

(A1)	(A2)	(A3)	(A5)	(A6)
Block name	**Fonction description**	**Failure Mode**	**Failure mode FIT** (only to verify faire rate distribution)	**Undesired event provoked? (directly or not)**

Note : *The template represented here may slightly differ from the one that is currently used by Valeo, it is based on an older version of the qualitative FMEDA.*

The next table describes the quantitative FMEDA header. Quantitative FMEDA determine the way with which of each failure of a block contribute to a considered UE in order to calculate the local architectural metrics for this component UE. This is the description of the different fields composing it:

- **(B1):** potential failure mode of the functional block (corresponds to the column A4 in the qualitative FMEDA header),
- **(B2):** failure rate of the contributing failure mode,
- **(B3):** boolean that tells if the failure modes can directly cause the occurrence of the undesired event,
- **(B4):** this field contains the name of the first order safety mechanism associated to the failure mode (if existing),
- **(B5):** diagnostic coverage of the first order mechanism,

- **(B6):** residual failure rate (uncovered) that directly contributes to the occurrence of an undesired event,
- **(B7):** boolean that tells if a failure mode can lead to the occurrence of the unwanted event in combination of other failure modes ,
- **(B8):** name of the safety mechanism that avoid this failure mode from being latent,
- **(B9):** proportion of failure rate that is doesn't remain latent for this failure mode,
- **(B10):** latent failure rate (undetected) of the failure mode,
- **(B11):** Boolean; tells if we consider the failure mode as safety related or not.

Table 5:4 Quantitative FMEDA header

(B1)	(B2)	(B3)	(B4)	(B5)	(B6)	(B7)	(B8)	(B9)	(B10)	(B11)
Failure Mode	**contributing lambda**	**Failure mode that has the potential to violate the safety goal in absence of Safety Mechanisms?**	**Safety mechanism(s) allowing to prevent the failure mode from violating the safety goal?**	**Failure mode coverage wrt. Violation of safety goal**	**Residual or Single Point Fault failure rate / FIT**	**Failure mode that may lead to the violation of safety goal in combination with an independent failure of another component?**	**Detection means? Safety mechanism(s) allowing to prevent the failure mode from being latent**	**Failure mode coverage wrt. Latent failures**	**Latent Multiple Point Fault failure rate / FIT**	**Safety Related ?**

The complete FMEDA can be built, by performing a "join" operation between the two previously presented tables, based on the field failure mode (respectively A4 and B1 in the two previous tables)

5.4.2 Qualitative FMEDA Coherence regarding Fault Trees Report

The Qualitative FMEDA is one of the first activities done during the safety analyses. One of its main characteristics is that it allows reaching a good level of exhaustiveness for the elicitation of the different failure modes that can attain an assessed component.

It is of great interest to be able to use this exhaustiveness to verify that our Fault Trees are complete and contain all the basic events that could lead to the violation of a safety goal.

To do so, we try to associate each line of the Qualitative FMEDA with a Fault Tree basic Event.

The next step is comparing each basic event name of the fault tree to the failure modes. From this comparison, we can obtain three results:

- If the two fields match, the failure mode identified in the Qualitative FMEDA is implemented in the Fault Tree. We can thus perform a check on the fields (A5) and (A6):
  - A5 should corresponds to the failure rate associated to the basic event,
  - A6 should corresponds to the top gate of the fault tree;

- If a failure mode doesn't find its match in the basic events names, this can be due to two different reasons:
    - either the failure mode does not contribute to the occurrence of the assessed undesired event, and thus it's corresponding field in (A5) should be set to a different value than the top gate event in the fault tree,
    - or, it is not implemented properly in the fault tree.

  In both cases, we can add this failure mode name to a list, and provide it to the analyst to help him either complete the fault tree if necessary, or check that the listed failure modes don't contribute to the UE.
- If a Basic Event name doesn't find its match in the Failure Modes, this can also be due to two reasons:
    - the first possible reason has been deducted after an assessment of various Qualitative FMEDAs performed by Valeo: we observed that in general the Second Order Safety Mechanisms are not introduced into the FMEDA until the realization of the quantitative part. Thus if an element is tagged as second order safety mechanism, then it absence can be explained.
    - the other possible reason, is that the basic event, was not identified during the Qualitative FMEDA or that it has not been correctly implemented regarding to the Failure Mode that should correspond to it (due for example to typography differences).

  In both cases, we can add this Basic Event name to a list, and provide it to the safety expert in order to help him either complete the Qualitative FMEDA or the Fault Tree.

Also, in order to help the analysts to identify the mismatches that could be due to small misspelling or typography errors, we implemented an algorithm for the calculation of Jaro-Winkler distance between the elements of each list. This distance allows comparing two characters strings giving a result between 0 and 1. The more the strings are similar the higher is the result. Of courses other algorithms could be implemented as for example the classic Levenshtein distance (Winkler, 2006).

Thus, we can compare each element of the list of unmatched Failure Modes with the list of unmatched Basic Events, and check there are a couple with a sufficiently high distance value.

```
WARNING : Unmatched Failure modes in the quantivative FMEDA (may be not safety related) :
   Output commanded when normaly not
   Loss of watchdoge (SM4 - Monitoring of uC program execution)
   Permanent Reset
   No Safety Signalisation (Command Lamp)
   Erroneous Temperature Signal
   Loss of Temperature Signal
   Valve 2 never commanded
   Valve 2 always commanded
   Erroneous Valve 2 state reading (SM1 - Monitoring of output stage controlling T71)
   Loss of Valve 2 state reading (SM1 - Monitoring of output stage controlling T71)
   Valve 1 never commanded
   Erroneous Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)


WARNING : Unmatched Basic Event in the Fault Tree (may be a second order safety mechanism) :
   Loss of watchdog (SM4 - Monitoring of uC program execution)
   Loss of SM5
   ESD Overvoltage


HELP : Possible close matches list :
   Loss of watchdoge (SM4 - Monitoring of uC program execution) --------- Loss of watchdog (SM4 - Monitoring of uC program execution)
---BYE---
```

Figure 5:10 Generated Coherence Report Examples for a Slightly Modified ISO26262 Part 5 Annex E SG02 Safety Analyses

In Figure 5:10, we can observe an example of report generated from a comparison between qualitative FMEDA to its corresponding Fault Tree. This is based on the safety goal example described in ISO26262 part 5, Annex E. More details about the input data will be given in Annex A of this document.

5.4.3 Quantitative FMEDA Generation from Tagged Fault Trees

The Quantitative FMEDA is a safety analysis that is generally made in parallel of a fault tree analysis. It contains architectural information and numeric data. These data allow representing the way that each analyzed failure mode contributes to the occurrence of an unwanted event.

As all this data are already contained in our tagged fault trees, their generation directly from fault trees could be of a great help to assist the analysts. To do this, a tagged fault tree must be processed using a certain strategy. In the following we detail the process that allowed us to generate our quantitative FMEDA from tagged fault trees using coverage gate.

First of all, from the list of all basic events we generate a set of n-tuples named ***originalBE***. We compose them as following ***(basicEventName, failureRate, associatedSafetyMechanisms, coverage, coefficient, dependence, eventType)***. We initialize each parameter as the following:

- ***basicEventName:*** Name of the basic event that this n-tuple is created from;
- ***failureRate:*** Its corresponding failure rate;
- ***associatedSafetyMechanisms:*** Empty at the initialization, it is intended to be filled with the name of the Safety Mechanisms that covers this Basic Event;
- ***coverage:*** Empty at the initialization, it is intended to contains the resulting coverage of the safety mechanisms that covers this Basic Event;
- ***coefficient:*** If it exists, it shall contain the perception or signalization rate of the basic event;
- ***dependence:*** is a Boolean variable which is set to true if this Basic Event has a signalization rate. The role of this parameter is to tell if a safety mechanism is responsible of the perceivability of this Basic Event;
- ***eventType:*** Corresponds to the type of event represented in the fault tree: (“normal”, “first-order”, “second-order”).

Next to this, we create a set of n-tuples named originalGates and composed as following ***(gateName, associatedSafetyMechanisms, coverage)***. We initialize each parameter as the following:

- ***gateName:*** Name of the gate that this n-tuple is created from;
- ***associatedSafetyMechanisms:*** Empty at the initialization, it is intended to be filled with the name of the name Safety Mechanisms that covers this Basic Event;
- ***coverage:*** Empty at the initialization, it is intended to contains the resulting coverage of the safety mechanisms that covers this Basic Event.

Now, starting from the top Gate Event we proceed as following:

- If the currently selected gate is an Or Gate, we get the *associatedSafetyMechanisms* and *coverage* from its n-tuple, and propagate their values to its sub elements;
- If the currently selected gate is an And Gate, we get the *associatedSafetyMechanism* and *coverage* from its n-tuple, than we propagate their values to the first element of the And Gate (to be coherent with 5.2);
- If the currently selected gate is a Coverage Gate :
  - we get the name of its second sub element (safety mechanism of the coverage gate) and combine it with the value of *associatedSafetyMechanism* from its n-tuple and propagate it to its first sub element;
  - if the *coverage* of that n-tuple is empty, then we propagate the value of the third parameter of the Coverage Gate (diagnostic coverage *DC* of the safety mechanism of the gate), else we propagate the following value: $1 - (1 - coverage)(1 - DC)$.

We iterate these steps on each sub element of the selected gate and their sub elements… etc., as long as it is possible.

The propagation operation previously mentioned consists in updating the n-tuples contained in *originalGates* and *originalBE*, by affecting the transmitted values to their *associatedSafetyMechanism* and *coverage* parameters.

From *originalBE*, we create two subsets:

- if a basic event is a descendant of an and Gate or the second sub element of a Coverage Gate, or is himself this sub element, then we affect his tuple into a set that we named *CombinedBE* (each basic event corresponding to a tuple in this subset is an event that can never lead to the top event on his own);
- Else, we affect the tuple to a set that we named *DirectBE* (each of the basic events corresponding to the n-tuples in this subset can potentially directly lead to the triggering of the top tree event).

Using these two subsets, we can start the construction of the quantitative FMEDA: for each element in *DirectBE*, we generate a new line for our quantitative FMEDA with the following proprieties:

- Failure Mode **(B1)** is set to the value of *basicEventName*;
- Contributing Lambda **(B2)** is set to the value of *failureRate*;
- The potential to violate the safety goal in absence of safety mechanism **(B3)**, is set to true, as each Basic event represented by an element in *DirectBE* can directly lead to the occurrence of the top gate event (single or residual fault);
- If *associatedSafetyMechanisms* is not empty, we affect its value to **(B4)** which represent the first order safety mechanisms associated to this Failure Mode;
- If *coverage* is not empty, we affect its value to **(B5)** to represent the diagnostic coverage of the first order safety mechanisms over the failure mode which correspond to this line;
- The residual single point fault rate **(B6)** corresponds to the following value : $failureRate \times (1 - coverage)$;
- If there is at least a safety mechanism (*associatedSafetyMechanisms* not empty), then **(B7)** is set to true, as the failure of the safety mechanism combined with the basic event occurrence can lead to the occurrence of the Top Event;
- If *dependence* is set to true then we affect the value of *associatedSafetyMechanisms* to **(B8)**, as it allows signaling the occurrence of the current failure mode;
- We affect the value of coefficient to **(B9)** to represent the percentage of perception/signalization of the failure mode when it is latent.
- The latent failure rate **(B10)** corresponds to the following value : $\big((B4) - (B6)\big) \times (1 - (B9))$
- And finally, **(B11)** is set to true, as every event in a fault tree is considered safety related.

For each element in *CombinedBE* we generate a line following these rules:

- Failure Mode **(B1)** is set to the value of *basicEventName*;
- Contributing Lambda **(B2)** is set to the value of *failureRate*;
- We set **(B11)** to true, as every event in a fault tree is considered safety related.
- If the *eventType* is not set to *"second-Order"* (the event doesn't correspond to a second order safety mechanism failure) then :
    - The potential to directly violate the safety goal in absence of safety mechanism **(B3)**, is set to *false* (or let empty), as each Basic event represented in C*ombinedBE* needs at least to be combined with another event in order to lead to the top gate event occurrence;
    - The residual single point fault rate **(B6)** is set to zero, as neither of the events in *CombinedBE* can directly lead to the violation of the safety goal;

- The potential to violate a safety goal in combination with other events **(B7)** is set to true by construction;
- If *dependence* is set to true, we affect the value of *associatedsafetymechanisms* to **(B8)** which represent the first order safety mechanisms associated to this Failure Mode in order to avoid its latency;
- If dependence is set to true, we affect the value of $(coefficient)$ to **(B9)**;
- The latent failure rate **(B10)** corresponds to the following value : $(B4) \times \big(1 - (B9)\big)$;

- Else, in the case where the *eventType* is set to *"second-Order"*, then all the unset fields can stay empty or considered at their default values (false and zero).

As we can see, the data that are contained in our fault trees are sufficient to generate the Quantitative FMEDA lines that correspond to the template given in table 5:4. Also, this table content perfectly corresponds to the one that allowed us architectural metrics calculation in the previous section, so the calculated metrics could be directly associated to the FMEDA.

In Annex A, we can see the example of Quantitative FMEDA Generated based on a slightly modified fault tree corresponding to the second safety goal case defined in ISO 26262, Part 5, Annex E.

5.4.4 Complete FMEDA Generation

As previously indicated, the complete FMEDA can be constructed by performing a join operation between the Quantitative FMEDA and the Qualitative by performing a join operation between the two corresponding tables using the data contained in the column (A3) and (B1) of the presented templates.

So the steps to generate a complete FMEDA table using our custom Fault Trees and the Qualitative FMEDA are the following:

- First we must check the coherence between the assessed fault tree and the quantitative FMEDA :
  - We generate a coherence report using these two elements as input;
  - If there are incoherence or missing data, these must be corrected or completed;
- Then, we generate the Quantitative FMEDA corresponding to the fault tree;
- And finally we join the two tables in order to generate a complete FMEDA.

In order to complete the FMEDA analysis role, we can join to this, the result of the architectural metrics that are directly obtained from the corrected custom fault tree.

5.5 About the Implementation

The tool-chain described in this chapter was implemented by using Python scripts to build a prototype. The length of four of those scripts is comprised between 200 and 500 lines. The current implementation is sufficient when it comes to the processing of real size examples. However it may not be practical as it is lacking of a user-friendly graphical interface. In practice, the deployable version language will mostly depend on the tools environment and its interfacing abilities.

5.6 Conclusion

In this chapter, we presented some ISO 26262 specific developments that allowed us to assist and simplify a good part of the safety analyses that are performed by the automotive safety engineers.

We first presented a custom fault tree pattern and tags that allows to accurately represent the elements that are taken into account during the safety analyses (basic events, coverage, safety mechanisms); this can also be seen as a step forward in term of model based approach ; the introduced pattern represent each element distinctly with its own dependent parameters.

Next to this, and based on these custom fault trees, more classic fault trees can be easily generated and architectural metrics can be computed using the presented methodology.

Also, we build tools allowing to verify the coherence between theses fault trees and a Qualitative FMEDA, allowing more integrity in our safety analyses. We also constructed tools allowing to generate the Quantitative FMEDA from those fault trees and thus allowing the construction of complete FMEDA analyses.

Other little improvement could be easily implemented, as an auto-check in the coherence of the fault trees using a custom pattern: A basic event tagged with the “signaled” tag shall be a descendant of a Coverage Gate; a “second order safety mechanism” tagged basic event shall never be covered by another safety mechanism… etc. However, we preferred to let those decisions to the safety engineers in order to be sure to not restrain them.

Another improvement that could be easily implementable would be to extract data directly from the electronic FMEAs in order to compute the exact value of the architectural metrics.

The next chapter presents the first step of works that are more model-based. This was done in order to verify their feasibility investigate a possible the transition to high level safety analyses assessment.

CHAPTER 6

TOWARD A MODEL BASED GENERATION OF SAFETY ANALYSES

Chapter 6 Toward a model based generation of the safety analyses

In this chapter, we propose generic AltaRica 3 for Electric and Electronic Systems protected by first and second order safety mechanisms. These models are of a great help to clarify the behaviour of these systems as well as to determine the domain of validity of simpler models such the above mentioned Fault Trees or ad-hoc formulas.

6.1 AltaRica 3.0 Models

In this section we present AltaRica 3 models for the representation of automotive safety mechanisms. One of the main objectives behind this is to be able to define generic classes for the different objects that are handled during the safety analyses and which allow the representation of their dysfunctional behavior.

AltaRica 3 is the latest evolution of event based modelling language “AltaRica” (Prosvirnova, et al., 2013). In this Language, the state of the system is described by means of variables, so, the modification of the system state can only happen when its variables values change. Also, the value of these variable can only occurs with the occurrence of an event.

Events can be associated with deterministic or stochastic delays. Models of components can be assembled into hierarchies, their inputs and outputs can be connected and their transitions can be synchronized.

In the following sub sections, we will use the previously presented examples (chapter 3), to explain our implementation of the automotive safety related component.

6.2 AltaRica 3 Models for the Vehicle Management Unit for Inversion

In this subsection, we will describe, element by element how to represent the failure behaviour of the VMU for inversion.

So, to begin, each of the considered objects (functional blocks and safety mechanisms) can have two states: working and failed. So, we create the corresponding variable type:

```
domain HardwareStatus{WORKING, FAILED}
```

We also know that each object can be subject to failure when it is working, and is considered as repaired after maintenance. Both maintenance and failure are considered as events, and are represented in our models by transitions with the same respective name.

With this in mind, we can already obtain the AltaRica model for a generic Hardware Block:

```
class HardwareBlock
  HardwareStatus self (init = WORKING);
  Boolean failed (reset = false);
```

```
  Boolean failureDetected (reset = false);
  Boolean safeMode (reset = false);
  event failure (delay = exponential(lambda));
  event maintenance;
  transition
    failure : self == WORKING -> self := FAILED;
    maintenance : true -> self := WORKING;
  assertion
    failed := self == FAILED;
    safeMode := failed and failureDetected;
end
```

We also have three Boolean variables for external communication:

- "failed", that allows to read the state of the hardware block (for examples In VMU, when the refresh orders are not sent to the watch dog, this variable is set to "true").
- "safeMode", that allows to see if the hardware block is in safe mode. As defined in Section, the safe mode is engaged when the hardware block has failed ("failed" set to true) and its failure is contained by the first order safety mechanisms ("failureDetected" set to true).
- "failureDetected", is an input variable that allows to indicate if a failure is currently detected by the first order safety mechanism (this correspond to the restart order sent order in the VMU example).

In addition to those, a detection based first order safety mechanism must be able to detect and contain the failure of its associated hardware block.

In order to take into account the diagnostic coverage and its associated probabilities of detection we create two concurrent instantaneous transitions: One for the good detection of the hardware block failure and another for its wrong detection.

```
domain FaultStatus{DETECTED, PROPAGATION, NONE}
class DetectionBasedSafetyMechanism
  FaultStatus faultStatus (init = NONE);
  HardwareStatus self (init = WORKING);
  Boolean failed(reset = false);
  Boolean inputSignal (reset = false);
  Boolean sendSafeModeOrder (reset = false);
  event failure (delay = exponential(lambda));
  event goodDiagnostic(delay = 0, expectation = DC);
  event wrongDiagnostic(delay = 0, expectation = 1 - DC);
  event maintenance;
  transition
    failure : self == WORKING ->
        {faultStatus := PROPAGATION; self := FAILED;}
    goodDiagnostic: self == WORKING and inputSignal and
        faultStatus == NONE -> faultStatus := DETECTED;
    wrongDiagnostic: self == WORKING and inputSignal and
        faultStatus == NONE -> faultStatus := PROPAGATION;
    maintenance: true -> {faultStatus := NONE;
        self := WORKING;}

  assertion
```

```
    failed := self == FAILED;
    sendSafeModeOrder := faultStatus == DETECTED;
end
```

We consider that the first order safety mechanisms allow fault propagation in two cases: either the fault has not been detected, or, the safety mechanism has failed so it is impossible to stop the hardware block failure propagation.

By following the same logic, we can design the second order mechanism behaviour:

```
domain SignalisationStatus {SIGNALED, NONE}
class SecondOrderSafetyMechanism
  FaultStatus faultStatus (init = NONE);
  HardwareStatus self (init = WORKING);
  SignalisationStatus signalisationStatus (init = NONE);
  Boolean inputSignal(reset = false);
  event failure (delay = exponential(lambda));
  event goodDiagnostic(delay = Td, expectation = DC);
  event wrongDiagnostic(delay = Td, expectation = 1 - DC;
event maintenance;
  transition
    failure : self == WORKING -> self := FAILED;
goodDiagnostic: self == WORKING and
      faultStatus == NONE and inputSignal ->
      signalisationStatus := SIGNALED;
    wrongDiagnostic: self == WORKING and
         faultStatus == NONE and inputSignal -> skip;
    maintenance: true -> {self := WORKING;
         signalisationStatus := NONE;}
end
```

The main difference between this second order safety mechanism and the previously defined first order safety mechanism are the following:

- The second order safety mechanism do not stop the propagation of faults, it only signals them, and so, we consider that as long as a fault is signaled, it stays in this status even if the second order safety mechanism fails.
- The diagnostic coverages events, are periodically timed and not instantaneous as presented in the detection based first order safety mechanism. This represents the fact that the tests that only occurs at the vehicle start.

Now that we have the classes that correspond to each element in our system, we can combine them in order to represent the failure behaviour of our vehicle management unit (VMU).

```
class VMU
  Microcontroller uc;
  DetectionBasedSafetyMechanism watchDog;
  event repairRequestedBySM1 (delay = tTau);
  event repairRequestedBySM2 (delay = tTau);
  observer HardwareStatus status =
     if(uc.torqueCalculationFunction.failed
```

```
      and watchDog.faultStatus == PROPAGATION)
      then FAILED
      else WORKING;
  transition
    repairRequestedBySM1:
          watchDog.faultStatus==DETECTED -> skip; &
          !uc.torqueCalculationFunction.maintenance &
          !watchDog.maintenance &
          !uc.periodicTestingUnit.maintenance;
    repairRequestedBySM2:
         uc.periodicTestingUnit.signalisationStatus ==
         SIGNALED and
         not uc.torqueCalculationFunction.failed -> skip; &
         !uc.torqueCalculationFunction.maintenance &
         !watchDog.maintenance &
         !uc.periodicTestingUnit.maintenance;
    hide uc.periodicTestingUnit.maintenance,
         watchDog.maintenance,
         uc.torqueCalculationFunction.maintenance;
  assertion
    watchDog.inputSignal :=
        uc.torqueCalculationFunction.failed;
    uc.periodicTestingUnit.inputSignal :=
        watchDog.failed;
    uc.torqueCalculationFunction.failureDetected :=
        watchDog.sendSafeModeOrder;
end
```

As detailed in chapter 3, the VMU is composed by the microcontroller and a watchdog. The Microcontroller is in charge of two functions: The torque calculation function and the periodic testing of the watchdog. We consider that the VMU has failed only when the torque calculation function has failed and when the watchdog can't contain the propagation.

Also, there are two events that lead to maintenance: either the failure of the torque function is detected by the watchdog or the failure of the watchdog is detected by the second order safety mechanism. This is represented respectively by the transitions "repairRequestedBySM1" and "repairRequestedBySM2".

6.3 AltaRica 3 Models for Electric Driver Seat Control

In this subpart, we present the modifications that are necessary in our previous model in order to be able to represent systems and components with first order safety mechanisms based on inhibition.

The main difference between this model and the previous one is in the type of the first order safety mechanism that is used. Indeed, as described in Chapter 3. This system implements a safety mechanism based on inhibition.

First of all, to realize this implementation, we consider that as long as the first order safety mechanism is active then the associated hardware block is not powered. Thus we change the failure transition of our hardware block by adding this condition:

```
failure : self == WORKING and powered -> self := FAILED;
```

We also must define the safety mechanism based on inhibition:

```
class InhibitionBasedSafetyMechanism
  HardwareStatus self (init = WORKING);
  Boolean failed (reset = false);
  Boolean functionInhibition (init = true);
  event failure (delay = exponential(lambda));
  event maintenance;
  transition
    failure : self == WORKING ->
        functionInhibition := false; self := FAILED;
    maintenance : true ->
        functionInhibition := true; self := WORKING;
  assertion
    failed := self == FAILED;
end
```

As we can see, as long as this safety mechanism is working, it forces the function inhibition by setting its Boolean communication output “functionInhibition” to the value “true”.

For the second order safety mechanism behaviour implementation, we use the one that was presented in the previous section.

To finish this implementation, we created the class EDSC that is in charge of combining the previously presented components:

```
class EDSC
  HardwareBlock driverSeatsManager;
  InhibitionBasedSafetyMechanism powerInhibition;
  SecondOrderSafetyMechanism periodicTestingUnit;
  event repairRequestedBySM2 (delay = tTau);
  observer HardwareStatus status =
    if (driverSeatsManager.failed) then FAILED
    else DEFAULT;
  transition
    repairRequestedBySM2:
      periodicTestingUnit.signalisationStatus == SIGNALED
      and not driverSeatsManager.failed -> skip; &
      !powerInhibition.maintenance &
      !driverSeatsManager.maintenance &
      !periodicTestingUnit.maintenance;

    hide periodicTestingUnit.maintenance,
         driverSeatsManager.maintenance,
         powerInhibition.maintenance;
  assertion
    periodicTestingUnit.inputSignal :=
              powerInhibition.failed;
    driverSeatsManager.powered :=
              not powerInhibition.functionInhibition;
end
```

As we can see, the only condition for the system failure is that that the driver seat manager fails. But as the different objects were designed, this event can happen only after the failure of the inhibition based first order safety mechanism.

6.4 Reachability Graphs

Using a stepper, we built the reachability graph for each of the presented AltaRica models.

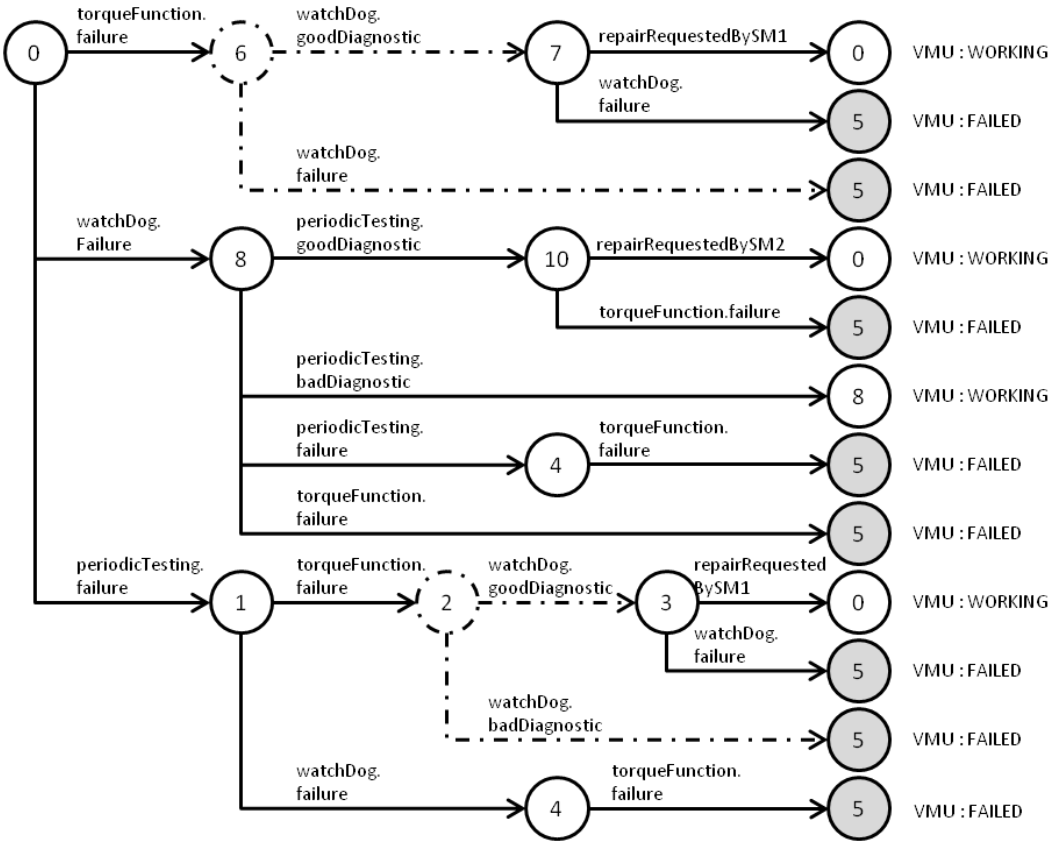

Figure 6:1 Reachability graph of the VMU AltaRica 3 Model matched with the unfolded view of the corresponding Markov Model.

As we can see in Figure 6:1, our VMU AltaRica model perfectly matches the Markov model presented in Figure 3:4. The states numbers in this graphic are the same than the ones presented in the corresponding Markov model. This allows us to say that AltaRica model presented in Section 6.2 provides good implementations of the automotive Safety Mechanisms.

The same work has been done to compare the model proposed in Section 6.3 with the one illustrated in Figure 3:5. In this case too, the two models perfectly match, allowing us to deduce that they provide good implementations of Automotive Safety Mechanisms.

6.5 Conclusion

In this chapter, we proposed AltaRica 3 models for the behavior of large classes of automotive Electric and Electronic systems protected by first and possibly second order safety mechanisms. These models are generics in the sense that only few adjustments must be made in order to use them in most of the practical cases.

The modularity of the models that we proposed allows to easily combining them in order to represent large sized systems. Also, given the fact that they can be compiled to Guarded Transition Systems models, the computations on these models can be really fast and accurate (by using the Limited Depth Markov Generation (Brameret, et al., 2013) (Brameret, et al., 2015)).

CHAPTER 7

CONCLUSION

Chapter 7 Conclusion

The results described in these manuscripts were the result of three years of collaboration between Valeo and Ecole Polytechnique.

One of the main objectives that we had to consider during this project was to provide Valeo with concrete solutions for enhancing their safety analyses process.

Each chapter can be seen as an independent step bringing us closer to our objective: automotive safety analyses enhancement and efficient generation.

The interest of the work performed during this thesis was threefold:

First, the Markov models proposed here were of a great help to clarify and formalize the behavior of the automotive safety mechanisms. It also allowed us to determine the domain of validity of their main parameters.

This allowed us to provide accurate fault trees approximations. The determination and the study of Fault Tree patterns is of a special interest for most of the analysts which are familiar with this technology, as they allows a convenient manipulation and representation of the various event that can lead to a failure.

The patterns deducted from these fault trees allowed us to build robust ISO 26262 specific processes for metrics calculations and safety analyses generation.

Secondly, the work performed here helped us having a better understanding of ISO 26262 standards: during the assessment of the Valeo safety methodology and ISO26262 standard, with the help of functional safety experts and practitioners, we identified some points that should be either enhanced or clarified within the standard.

Currently, a part of this thesis results are already in use within Valeo: the work that were performed on the fault trees are currently implemented in a guideline for fault tree analyses.

The scripts developed for Quantitative FMEDA generation and coherence check are currently being integrated in the Valeo Safety process and should be deployed in the next few months.

And finally, designing efficient AltaRica 3 classes helped us to prepare the investigation of high level model based safety assessment.

ANNEX A
FMEDA GENERATION EXAMPLE

Annex A FMEDA Generation Example

In this annex, we present an example for FMEDA Generation based on the “safety goal 2” example which is given in ISO26262 part 5 annex E.

The safety goal 2 violation corresponds to the following event : “Valve1 closed for longer than Y ms when Speed is higher than 100kph”

For more completeness, we slightly modified this example in order to include a Second Order Safety Mechanism (called SM5 in the following fault trees).

In the next section, we first present the fault tree corresponding to the unwanted corresponding to this safety goal and the quantitative FMEDA associated to it. Then, we present the various outputs that are generated in order to show the Quantitative FMEDA generated, the metrics calculated, and the coherence report generated.

Note: *The Qualitative FMEDA presented in this Annex contains some orthographic and typographic faults; these were not corrected in order to* ***preserve the example authenticity****.*

So, instead of modifying the qualitative FMEDA, the “consistency-check” module of our toolchain allowed us to introduce these faults in our FTA in order to be able to generate the Quantitative FMEDA.

A.1 Inputs Descriptions

A.1.1 Qualitative FMEDA

Block name	Fonction description	Failure Mode	Failure mode FIT (only to verify faire rate distribution)	Undesired event provoked?
Block 01	Data Treatement	Output commanded when normaly not	50	SG2 violation
Block 01	Data Treatement	Output not commanded when normaly commanded	50	
Block 02	Watch Dog - Monitoring of uC program execution (SM4)	Loss of watchdog (SM4 - Monitoring of uC program execution)	10	
Block 02	Watch Dog - Monitoring of uC program execution (SM4)	Permanent Reset	10	SG2 violation
Block 03	Command Lamp	No Safety Signalisation (Command Lamp)	12	SG2 violation
Block 11	Temperature Aquisition	Erroneous Temperature Signal	1.1	SG2 violation
Block 11	Temperature Aquisition	Loss of Temperature Signal	3.3	SG2 violation
Block 12	Command Valve 2	Valve 2 never commanded	12.7	SG2 violation

Block 12	Command Valve 2	Valve 2 always commanded	2.5	SG2 violation
Block 13	Monitoring of output stage controlling T71 (SM1)	Erroneous Valve 2 state reading (SM1 - Monitoring of output stage controlling T71)	2	SG2 violation
Block 13	Monitoring of output stage controlling T71 (SM1)	Loss of Valve 2 state reading (SM1 - Monitoring of output stage controlling T71)	1.8	SG2 violation
Block 21	Wheel Speed Aquisition 1	Erroneous Wheel Speed signal 1	2.8	
Block 21	Wheel Speed Aquisition 1	Loss of Wheel Speed signal 1	8.6	
Block 22	Wheel Speed Aquisition 2	Erroneous Wheel Speed signal 2	2.8	
Block 22	Wheel Speed Aquisition 2	Loss of Wheel Speed signal 2	8.6	
Block 23	Comparison-Consistency check of Wheel Speed inputs (SM2)	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	2	
Block 24	Command Valve 1	Valve 1 never commanded	12.7	SG2 violation
Block 24	Command Valve 1	Valve 1 always commanded	2.5	
Block 100	ESD - Voltage Overload	Loss of ESD - Electrical protection	1	SG2 violation
Block 25	Monitoring of output stage controlling T61 (SM3)	Erroneous Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)	2	SG2 violation
Block 25	Monitoring of output stage controlling T61 (SM3)	Loss of Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)	1.8	

A.1.2 Fault Tree with Coverage Gates

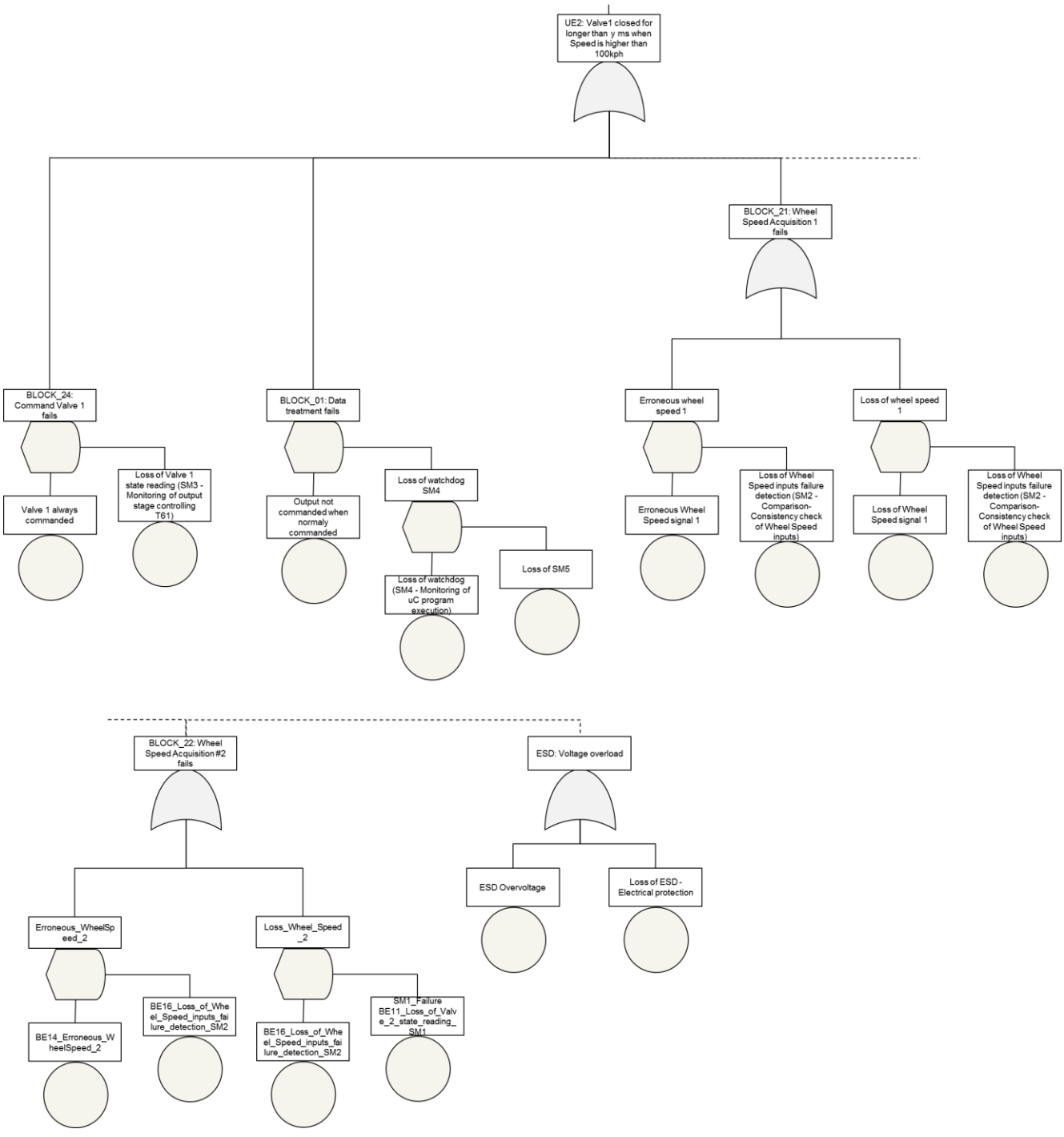

Figure A:1ISO 26262 Annex E SG2 coverage gates fault tree

A.1.2.1 Coverage Fault Tree OpenPSA Description

```
<?xml version="1.0" ?>
<!DOCTYPE opsa-mef>
<opsa-mef>
	<define-fault-tree name="FT Part 5 SG2">
		<define-gate name="TOP">
			<or>
				<gate name="BLOCK_24: Command Valve 1 fails"/>
				<gate name="BLOCK_01: Data treatment fails"/>
				<gate name="BLOCK_21: Wheel Speed Acquisition 1 fails"/>
				<gate name="BLOCK_22: Wheel Speed Acquisition 2 fails"/>
				<gate name="ESD: Voltage overload"/>
			</or>
		</define-gate>
		<define-gate name="BLOCK_24: Command Valve 1 fails">
			<coverage>
				<basic-event name="Valve 1 always commanded"/>
				<basic-event name="Loss of Valve 1 state reading (SM3 - Monitoring of
					output stage controlling T61)"/>
				<parameter name="DiagnosticCoverage3"/>
			</coverage>
		</define-gate>
		<define-gate name="BLOCK_01: Data treatment fails">
			<coverage>
				<basic-event name="Output not commanded when normaly commanded"/>
				<gate name="Loss of watchdog SM4"/>
				<parameter name="DiagnosticCoverage4"/>
			</coverage>
		</define-gate>
		<define-gate name="Loss of watchdog SM4">
			<coverage>
				<basic-event name="Loss of watchdog (SM4 - Monitoring of uC program
					execution)"/>
				<basic-event name="Loss of SM5"/>
				<parameter name="DiagnosticCoverage5"/>
			</coverage>
		</define-gate>

		<define-gate name="BLOCK_21: Wheel Speed Acquisition 1 fails">
			<or>
				<gate name="Erroneous wheel speed 1"/>
				<gate name="Loss of wheel speed 1"/>
			</or>
		</define-gate>
		<define-gate name="Erroneous wheel speed 1">
			<coverage>
				<basic-event name="Erroneous Wheel Speed signal 1"/>
				<basic-event name="Loss of Wheel Speed inputs failure detection (SM2 –
					Comparison-Consistency check of Wheel Speed inputs)"/>
				<parameter name="DiagnosticCoverage2"/>
			</coverage>
		</define-gate>

		<define-gate name="Loss of wheel speed 1">
			<coverage>
				<basic-event name="Loss of Wheel Speed signal 1"/>
				<basic-event name="Loss of Wheel Speed inputs failure detection (SM2 –
					Comparison-Consistency check of Wheel Speed inputs)"/>
				<parameter name="DiagnosticCoverage2"/>
			</coverage>
		</define-gate>

		<define-gate name="BLOCK_22: Wheel Speed Acquisition 2 fails">
			<or>
				<gate name="Erroneous wheel speed 2"/>
				<gate name="Loss of wheel speed 2"/>
			</or>
		</define-gate>
		<define-gate name="Erroneous wheel speed 2">
			<coverage>
				<basic-event name="Erroneous Wheel Speed signal 2"/>
				<basic-event name="Loss of Wheel Speed inputs failure detection (SM2 –
					Comparison-Consistency check of Wheel Speed inputs)"/>
				<parameter name="DiagnosticCoverage2"/>
```

```
        </coverage>
</define-gate>

<define-gate name="Loss of wheel speed 2">
        <coverage>
                <basic-event name="Loss of Wheel Speed signal 2"/>
                <basic-event name="Loss of Wheel Speed inputs failure detection (SM2 –
                        Comparison-Consistency check of Wheel Speed inputs)"/>
                <parameter name="DiagnosticCoverage2"/>
        </coverage>

</define-gate>
<define-gate name="ESD: Voltage overload">
        <and>
                <basic-event name="ESD Overvoltage"/>
                <basic-event name="Loss of ESD - Electrical protection"/>
        </and>
</define-gate>

<define-basic-event name="Valve 1 always commanded" type="Normal"
        signalisation-rate="1">
        <exponential>
                <parameter name="lambdaBE18"/>
                <mission-time />
        </exponential>
</define-basic-event>
<define-basic-event name="Loss of Valve 1 state reading (SM3 - Monitoring of output
        stage controlling T61)" Safety-Mechanism-Type="First Order">
        <exponential>
                <parameter name="lambdaBE20"/>
                <mission-time />
        </exponential>
</define-basic-event>

<define-basic-event name="Output not commanded when normaly commanded" type="Normal"
        signalisation-rate="1">
        <exponential>
                <parameter name="lambdaBE2"/>
                <mission-time />
        </exponential>
</define-basic-event>
<define-basic-event name="Loss of watchdog (SM4 - Monitoring of uC program
        execution)" Safety-Mechanism-Type="First Order" signalisation-rate="0.9">
        <exponential>
                <parameter name="lambdaBE3"/>
                <mission-time />
        </exponential>
</define-basic-event>
<define-basic-event name="Loss of SM5" Safety-Mechanism-Type="Second Order">
        <exponential>
                <parameter name="lambdaBESM5"/>
                <mission-time />
        </exponential>
</define-basic-event>

<define-basic-event name="Erroneous Wheel Speed signal 1" type="Normal"
signalisation-rate="1">
        <exponential>
                <parameter name="lambdaBE12"/>
                <mission-time />
        </exponential>
</define-basic-event>
<define-basic-event name="Loss of Wheel Speed inputs failure detection (SM2 –
Comparison-Consistency check of Wheel Speed inputs)"
Safety-Mechanism-Type="First Order">
        <exponential>
                <parameter name="lambdaBE16"/>
                <mission-time />
        </exponential>
</define-basic-event>
<define-basic-event name="Loss of Wheel Speed signal 1" type="Normal"
signalisation-rate="1">
        <exponential>
                <parameter name="lambdaBE13"/>
                <mission-time />
        </exponential>
```

```
</define-basic-event>

<define-basic-event name="Erroneous Wheel Speed signal 2" type="Normal"
signalisation-rate="1">
        <exponential>
                <parameter name="lambdaBE14"/>
                <mission-time />
        </exponential>
</define-basic-event>
<define-basic-event name="Loss of Wheel Speed signal 2" type="Normal"
signalisation-rate="1">
        <exponential>
                <parameter name="lambdaBE15"/>
                <mission-time />
        </exponential>
</define-basic-event>
<define-basic-event name="ESD Overvoltage" type="Normal">
        <exponential>
                <parameter name="lambdaBENull"/>
                <mission-time />
        </exponential>
</define-basic-event>
<define-basic-event name="Loss of ESD - Electrical protection" type="Normal">
        <exponential>
                <parameter name="lambdaBE22"/>
                <mission-time />
        </exponential>
</define-basic-event>

<define-parameter name="lambdaBE2" >
        <float value="50e-9" />
</define-parameter>
<define-parameter name="lambdaBE3" >
        <float value="10e-9" />
</define-parameter>

<define-parameter name="lambdaBE12" >
        <float value="2.8e-9" />
</define-parameter>
<define-parameter name="lambdaBE13" >
        <float value="8.6e-9" />
</define-parameter>
<define-parameter name="lambdaBE14" >
        <float value="2.8e-9" />
</define-parameter>
<define-parameter name="lambdaBE15" >
        <float value="8.6e-9" />
</define-parameter>
<define-parameter name="lambdaBE16" >
        <float value="2e-9" />
</define-parameter>

<define-parameter name="lambdaBE18" >
        <float value="2.5e-9" />
</define-parameter>
<define-parameter name="lambdaBE20" >
        <float value="1.8e-9" />
</define-parameter>
<define-parameter name="lambdaBESM5" >
        <float value="0" />
</define-parameter>
<define-parameter name="DiagnosticCoverage2" >
        <float value="0.99" />
</define-parameter>
<define-parameter name="DiagnosticCoverage3" >
        <float value="0.9" />
</define-parameter>
<define-parameter name="DiagnosticCoverage4" >
        <float value="0.9" />
</define-parameter>
<define-parameter name="DiagnosticCoverage5" >
        <float value="1" />
</define-parameter>
                <define-parameter name="lambdaBE22" >
        <float value="1e-9" />
</define-parameter>
```

```
		<define-parameter name="lambdaBENull" >
			<float value="0" />
		</define-parameter>
	</define-fault-tree>
</opsa-mef>
```

A.2 Outputs Descriptions

A.2.1 Generated And/Or Fault Tree OpenPSA Description

```
<?xml version="1.0"?>
<!DOCTYPE opsa-mef>
<opsa-mef>
	<define-fault-tree>
		<define-gate name="TOP">
			<or>
				<gate name="BLOCK_24: Command Valve 1 fails"/>
				<gate name="BLOCK_01: Data treatment fails"/>
				<gate name="BLOCK_21: Wheel Speed Acquisition 1 fails"/>
				<gate name="BLOCK_22: Wheel Speed Acquisition 2 fails"/>
				<gate name="ESD: Voltage overload"/>
			</or>
		</define-gate>
		<define-gate name="BLOCK_24: Command Valve 1 fails SM-Failure">
			<and>
				<basic-event name="Valve 1 always commanded (covered)"/>
				<basic-event name="Loss of Valve 1 state reading (SM3 - Monitoring of
				output stage controlling T61)"/>
			</and>
		</define-gate>
		<define-gate name="BLOCK_24: Command Valve 1 fails">
			<or>
				<basic-event name="Valve 1 always commanded (uncovered)"/>
				<gate name="BLOCK_24: Command Valve 1 fails SM-Failure"/>
			</or>
		</define-gate>
		<define-gate name="Loss of watchdog SM4 SM-Failure">
			<and>
				<basic-event name="Loss of watchdog (SM4 - Monitoring of uC program
				execution) (covered)"/>
				<basic-event name="Loss of SM5"/>
			</and>
		</define-gate>
		<define-gate name="Loss of watchdog SM4">
			<or>
				<basic-event name="Loss of watchdog (SM4 - Monitoring of uC program
				execution) (uncovered)"/>
				<gate name="Loss of watchdog SM4 SM-Failure"/>
			</or>
		</define-gate>
		<define-gate name="BLOCK_01: Data treatment fails SM-Failure">
			<and>
				<basic-event name="Output not commanded when normaly commanded
				(covered)"/>
				<gate name="Loss of watchdog SM4"/>
			</and>
		</define-gate>
		<define-gate name="BLOCK_01: Data treatment fails">
			<or>
				<basic-event name="Output not commanded when normaly commanded
				(uncovered)"/>
				<gate name="BLOCK_01: Data treatment fails SM-Failure"/>
			</or>
		</define-gate>
		<define-gate name="BLOCK_21: Wheel Speed Acquisition 1 fails">
			<or>
				<gate name="Erroneous wheel speed 1"/>
				<gate name="Loss of wheel speed 1"/>
			</or>
		</define-gate>
		<define-gate name="Erroneous wheel speed 1 SM-Failure">
```

```
        <and>
                <basic-event name="Erroneous Wheel Speed signal 1 (covered)"/>
                <basic-event name="Loss of Wheel Speed inputs failure detection (SM2 -
                 Comparison-Consistency check of Wheel Speed inputs)"/>
        </and>
</define-gate>
<define-gate name="Erroneous wheel speed 1">
        <or>
                <basic-event name="Erroneous Wheel Speed signal 1 (uncovered)"/>
                <gate name="Erroneous wheel speed 1 SM-Failure"/>
        </or>
</define-gate>
<define-gate name="Loss of wheel speed 1 SM-Failure">
        <and>
                <basic-event name="Loss of Wheel Speed signal 1 (covered)"/>
                <basic-event name="Loss of Wheel Speed inputs failure detection (SM2 -
                Comparison-Consistency check of Wheel Speed inputs)"/>
        </and>
</define-gate>
<define-gate name="Loss of wheel speed 1">
        <or>
                <basic-event name="Loss of Wheel Speed signal 1 (uncovered)"/>
                <gate name="Loss of wheel speed 1 SM-Failure"/>
        </or>
</define-gate>
<define-gate name="BLOCK_22: Wheel Speed Acquisition 2 fails">
        <or>
                <gate name="Erroneous wheel speed 2"/>
                <gate name="Loss of wheel speed 2"/>
        </or>
</define-gate>
<define-gate name="Erroneous wheel speed 2 SM-Failure">
        <and>
                <basic-event name="Erroneous Wheel Speed signal 2 (covered)"/>
                <basic-event name="Loss of Wheel Speed inputs failure detection (SM2 -
                Comparison-Consistency check of Wheel Speed inputs)"/>
        </and>
</define-gate>
<define-gate name="Erroneous wheel speed 2">
        <or>
                <basic-event name="Erroneous Wheel Speed signal 2 (uncovered)"/>
                <gate name="Erroneous wheel speed 2 SM-Failure"/>
        </or>
</define-gate>
<define-gate name="Loss of wheel speed 2 SM-Failure">
        <and>
                <basic-event name="Loss of Wheel Speed signal 2 (covered)"/>
                <basic-event name="Loss of Wheel Speed inputs failure detection (SM2 -
                Comparison-Consistency check of Wheel Speed inputs)"/>
        </and>
</define-gate>
<define-gate name="Loss of wheel speed 2">
        <or>
                <basic-event name="Loss of Wheel Speed signal 2 (uncovered)"/>
                <gate name="Loss of wheel speed 2 SM-Failure"/>
        </or>
</define-gate>
<define-gate name="ESD: Voltage overload">
        <and>
                <basic-event name="ESD Overvoltage"/>
                <basic-event name="Loss of ESD - Electrical protection"/>
        </and>
</define-gate>
<define-basic-event name="Valve 1 always commanded (covered)" signalisation-rate="1"
type="Normal">
        <exponential>
                <mul>
                        <parameter name="DiagnosticCoverage3"/>
                        <parameter name="lambdaBE18"/>
                </mul>
                <mission-time/>
        </exponential>
</define-basic-event>
<define-basic-event name="Valve 1 always commanded (uncovered)" type="Normal">
        <exponential>
                <mul>
```

```
                        <sub>
                                <int value="1"/>
                                <parameter name="DiagnosticCoverage3"/>
                        </sub>
                        <parameter name="lambdaBE18"/>
                </mul>
                <mission-time/>
        </exponential>
</define-basic-event>
<define-basic-event Safety-Mechanism-Type="First Order" name="Loss of Valve 1 state
reading (SM3 - Monitoring of output stage controlling T61)">
        <exponential>
                <parameter name="lambdaBE20"/>
                <mission-time/>
        </exponential>
</define-basic-event>
<define-basic-event name="Output not commanded when normaly commanded (covered)"
signalisation-rate="1" type="Normal">
        <exponential>
                <mul>
                        <parameter name="DiagnosticCoverage4"/>
                        <parameter name="lambdaBE2"/>
                </mul>
                <mission-time/>
        </exponential>
</define-basic-event>
<define-basic-event name="Output not commanded when normaly commanded
(uncovered)" type="Normal">
        <exponential>
                <mul>
                        <sub>
                                <int value="1"/>
                                <parameter name="DiagnosticCoverage4"/>
                        </sub>
                        <parameter name="lambdaBE2"/>
                </mul>
                <mission-time/>
        </exponential>
</define-basic-event>
<define-basic-event Safety-Mechanism-Type="First Order" name="Loss of watchdog
(SM4 - Monitoring of uC program execution) (covered)" signalisation-rate="0.9">
        <exponential>
                <mul>
                        <parameter name="DiagnosticCoverage5"/>
                        <parameter name="lambdaBE3"/>
                </mul>
                <mission-time/>
        </exponential>
</define-basic-event>
<define-basic-event Safety-Mechanism-Type="First Order" name="Loss of watchdog
(SM4 - Monitoring of uC program execution) (uncovered)">
        <exponential>
                <mul>
                        <sub>
                                <int value="1"/>
                                <parameter name="DiagnosticCoverage5"/>
                        </sub>
                        <parameter name="lambdaBE3"/>
                </mul>
                <mission-time/>
        </exponential>
</define-basic-event>
<define-basic-event Safety-Mechanism-Type="Second Order" name="Loss of SM5">
        <exponential>
                <parameter name="lambdaBESM5"/>
                <mission-time/>
        </exponential>
</define-basic-event>
<define-basic-event name="Erroneous Wheel Speed signal 1 (covered)"
signalisation-rate="1" type="Normal">
        <exponential>
                <mul>
                        <parameter name="DiagnosticCoverage2"/>
                        <parameter name="lambdaBE12"/>
                </mul>
                <mission-time/>
```

```
                </exponential>
            </define-basic-event>
            <define-basic-event name="Erroneous Wheel Speed signal 1 (uncovered)" type="Normal">
                <exponential>
                    <mul>
                        <sub>
                            <int value="1"/>
                            <parameter name="DiagnosticCoverage2"/>
                        </sub>
                        <parameter name="lambdaBE12"/>
                    </mul>
                    <mission-time/>
                </exponential>
            </define-basic-event>
            <define-basic-event Safety-Mechanism-Type="First Order" name="Loss of Wheel Speed
inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)">
                <exponential>
                    <parameter name="lambdaBE16"/>
                    <mission-time/>
                </exponential>
            </define-basic-event>
            <define-basic-event name="Loss of Wheel Speed signal 1 (covered)"
            signalisation-rate="1" type="Normal">
                <exponential>
                    <mul>
                        <parameter name="DiagnosticCoverage2"/>
                        <parameter name="lambdaBE13"/>
                    </mul>
                    <mission-time/>
                </exponential>
            </define-basic-event>
            <define-basic-event name="Loss of Wheel Speed signal 1 (uncovered)" type="Normal">
                <exponential>
                    <mul>
                        <sub>
                            <int value="1"/>
                            <parameter name="DiagnosticCoverage2"/>
                        </sub>
                        <parameter name="lambdaBE13"/>
                    </mul>
                    <mission-time/>
                </exponential>
            </define-basic-event>
            <define-basic-event name="Erroneous Wheel Speed signal 2 (covered)"
            signalisation-rate="1" type="Normal">
                <exponential>
                    <mul>
                        <parameter name="DiagnosticCoverage2"/>
                        <parameter name="lambdaBE14"/>
                    </mul>
                    <mission-time/>
                </exponential>
            </define-basic-event>
            <define-basic-event name="Erroneous Wheel Speed signal 2 (uncovered)" type="Normal">
                <exponential>
                    <mul>
                        <sub>
                            <int value="1"/>
                            <parameter name="DiagnosticCoverage2"/>
                        </sub>
                        <parameter name="lambdaBE14"/>
                    </mul>
                    <mission-time/>
                </exponential>
            </define-basic-event>
            <define-basic-event name="Loss of Wheel Speed signal 2 (covered)"
            signalisation-rate="1" type="Normal">
                <exponential>
                    <mul>
                        <parameter name="DiagnosticCoverage2"/>
                        <parameter name="lambdaBE15"/>
                    </mul>
                    <mission-time/>
                </exponential>
            </define-basic-event>
            <define-basic-event name="Loss of Wheel Speed signal 2 (uncovered)" type="Normal">
```

```
                <exponential>
                        <mul>
                                <sub>
                                        <int value="1"/>
                                        <parameter name="DiagnosticCoverage2"/>
                                </sub>
                                <parameter name="lambdaBE15"/>
                        </mul>
                        <mission-time/>
                </exponential>
        </define-basic-event>
        <define-basic-event name="ESD Overvoltage" type="Normal">
                <exponential>
                        <parameter name="lambdaBENull"/>
                        <mission-time/>
                </exponential>
        </define-basic-event>
        <define-basic-event name="Loss of ESD - Electrical protection" type="Normal">
                <exponential>
                        <parameter name="lambdaBE22"/>
                        <mission-time/>
                </exponential>
        </define-basic-event>
        <define-parameter name="lambdaBE2">
                <float value="50e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBE3">
                <float value="10e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBE12">
                <float value="2.8e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBE13">
                <float value="8.6e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBE14">
                <float value="2.8e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBE15">
                <float value="8.6e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBE16">
                <float value="2e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBE18">
                <float value="2.5e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBE20">
                <float value="1.8e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBESM5">
                <float value="0"/>
        </define-parameter>
        <define-parameter name="DiagnosticCoverage2">
                <float value="0.99"/>
        </define-parameter>
        <define-parameter name="DiagnosticCoverage3">
                <float value="0.9"/>
        </define-parameter>
        <define-parameter name="DiagnosticCoverage4">
                <float value="0.9"/>
        </define-parameter>
        <define-parameter name="DiagnosticCoverage5">
                <float value="1"/>
        </define-parameter>
        <define-parameter name="lambdaBE22">
                <float value="1e-9"/>
        </define-parameter>
        <define-parameter name="lambdaBENull">
                <float value="0"/>
        </define-parameter>
    </define-fault-tree>
</opsa-mef>
```

A.2.2 Tagged Basic Events and Minimal Cut Sets

Basic Event	Failure Rate	Tag
Valve 1 always commanded (covered)	2.25E-09	[signaled.100]
Valve 1 always commanded (uncovered)	2.50E-10	
Loss of Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)	1.80E-09	
Output not commanded when normaly commanded (covered)	4.50E-08	[signaled.100]
Output not commanded when normaly commanded (uncovered)	5.00E-09	
Loss of watchdog (SM4 - Monitoring of uC program execution) (covered)	1.00E-08	[signaled.90]
Loss of watchdog (SM4 - Monitoring of uC program execution) (uncovered)	0	
Loss of SM5	0	
Erroneous Wheel Speed signal 1 (covered)	2.77E-09	[signaled.100]
Erroneous Wheel Speed signal 1 (uncovered)	2.80E-11	
Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	2.00E-09	
Loss of Wheel Speed signal 1 (covered)	8.51E-09	[signaled.100]
Loss of Wheel Speed signal 1 (uncovered)	8.60E-11	
Erroneous Wheel Speed signal 2 (covered)	2.77E-09	[signaled.100]
Erroneous Wheel Speed signal 2 (uncovered)	2.80E-11	
Loss of Wheel Speed signal 2 (covered)	8.51E-09	[signaled.100]
Loss of Wheel Speed signal 2 (uncovered)	8.60E-11	
ESD Overvoltage	0	
Loss of ESD - Electrical protection	1.00E-09	

Minimal Cut Set		
Erroneous Wheel Speed signal 1 (uncovered)		
Erroneous Wheel Speed signal 1 (covered)	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	
Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	Loss of Wheel Speed signal 1 (covered)	
Erroneous Wheel Speed signal 2 (covered)	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	
Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	Loss of Wheel Speed signal 2 (covered)	
Loss of Wheel Speed signal 1 (uncovered)		
Erroneous Wheel Speed signal 2 (uncovered)		
Loss of Wheel Speed signal 2 (uncovered)		
Valve 1 always commanded (uncovered)		
Loss of Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)	Valve 1 always commanded (covered)	
Output not commanded when normaly commanded (uncovered)		
Loss of watchdog (SM4 - Monitoring of uC program execution) (uncovered)	Output not commanded when normaly commanded (covered)	
Loss of SM5	Loss of watchdog (SM4 - Monitoring of uC program execution) (covered)	Output not commanded when normaly commanded (covered)
ESD Overvoltage	Loss of ESD - Electrical protection	

A.2.3 Generated Quantitative FMEDA

Block name	Failure Mode	Safety Related lambda	Failure mode that has the potential to violate the safety goal in absence of Safety Mecha-nisms?	Safety mechanism(s) allowing to prevent the failure mode from violating the safety goal ?	Failure mode coverage wrt. Violation of safety goal	Residual or Single Point Fault failure rate / FIT	Failure mode that may lead to the violation of safety goal in combina-tion with an independent failure of another component?	Detection means? Safety mechanism(s) allowing to prevent the failure mode from being latent	Failure mode coverage wrt. Latent failures	Latent Multiple Point Fault failure rate / FIT	Safety Related ?
Block 01	Output not commanded when normaly commanded	5.00E-08	x	Loss of watchdog SM4	90%	5.00E-09	x	Loss of watchdog SM4	100%	0	x
Block 02	Loss of watchdog (SM4 - Monitoring of uC program execution)	1.00E-08					x	Loss of SM5	90%	1.00E-09	x
Block 21	Erroneous Wheel Speed signal 1	2.80E-09	x	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	99%	2.80E-11	x	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	100%	0	x
Block 21	Loss of Wheel Speed signal 1	8.60E-09	x	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	99%	8.60E-11	x	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	100%	0	x
Block 22	Erroneous Wheel Speed signal 2	2.80E-09	x	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	99%	2.80E-11	x	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	100%	0	x
Block 22	Loss of Wheel Speed signal 2	8.60E-09	x	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	99%	8.60E-11	x	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	100%	0	x

Block 23	Loss of Wheel Speed inputs failure detection (SM2 - Comparison-Consistency check of Wheel Speed inputs)	2.00E-09							x
Block 24	Valve 1 always commanded	2.50E-09	x	Loss of Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)	90%	2.50E-10	x	100%	x
Block 100	Loss of ESD - Electrical protection	1.00E-09			0%		x		x
Block 25	Loss of Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)	1.80E-09					x		x
Block 01	Output commanded when normaly not	0							
Block 02	Permanent Reset	0							
Block 03	No Safety Signalisation (Command Lamp)	0							
Block 11	Erroneous Temperature Signal	0							

Block 11	Loss of Temperature Signal	0									
Block 12	**Valve 2 never commanded**	**0**									
Block 12	**Valve 2 always commanded**	**0**									
Block 13	**Erroneous Valve 2 state reading (SM1 - Monitoring of output stage controlling T71)**	**0**									
Block 13	**Loss of Valve 2 state reading (SM1 - Monitoring of output stage controlling T71)**	**0**									
Block 24	**Valve 1 never commanded**	**0**									
Block 25	**Erroneous Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)**	**0**									

A.2.3 Generated Architectural Metrics and Coherence Report

```
SPFM = 93.92%  with L(single+residual) = 5.477999999999999e-09

LFM = 93.14%  with L(latent) = 5.8e-09

SR = 9.009999999999999e-08
```

```
WARNING : Unmatched Failure modes in the quantivative FMEDA (may be not safety related) :

   Output commanded when normaly not

   Permanent Reset

   No Safety Signalisation (Command Lamp)

   Erroneous Temperature Signal

   Loss of Temperature Signal

   Valve 2 never commanded

   Valve 2 always commanded

   Erroneous Valve 2 state reading (SM1 - Monitoring of output stage controlling T71)

   Loss of Valve 2 state reading (SM1 - Monitoring of output stage controlling T71)

   Valve 1 never commanded

   Erroneous Valve 1 state reading (SM3 - Monitoring of output stage controlling T61)


WARNING : Unmatched Basic Event in the Fault Tree (may be a second order safety mechanism) :

   Loss of SM5

   ESD Overvoltage

HELP : Possible close matches list :

   None

---BYE---
```

References

Amari, S., Myers, A., Rauzy, A. & Trivedi, K., 2008. Imperfect Coverage Models: Status and Trends. *Handbook of Performability Engineering,* p. 321–348.

Arberetier, E. & Brik, Z., 2010. System Dependability-Safety Modeling in Abstraction Levels. *LambdaMu 17.*

Avizienis, A., Laprie, J.-C., Randell, B. & Landwehr, C., 2004. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on,* 1(1), pp. 11-33.

Brameret, P.-A., Rauzy, A. & Roussel, J.-M., 2013. Preliminary System Safety Analysis with Limited Depth Markov Chain Generation. In Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems. *DCDS.*

Brameret, P.-A., Rauzy, A. & Roussel, J.-M., 2015. Automated generation of partial Markov chain from high level descriptions. *Reliability Engineering & System Safety,* Issue 139, pp. 179-187.

Elmqvist, J. & Nadm-Tehrani, S., 2007. Safety interface for component based systems. *Lecture Note For Computer Science (LNCS),* Volume 3688, pp. 246-260.

Elmqvist, J. & Nadm-Tehrani, S., 2008. Tool support for incremental FMEA of component based systems. *Design, Automation and Test in Europe,* pp. 921 - 927.

Epstein, S. & Rauzy, A., 2008. *Open-PSA Model Exchange format, version 2.0d,* s.l.: s.n.

Holub, P. & Börcsök, J., 2009. Advanced PFH calculations for safety integrity systems with high diagnostic. *Symposium on Information, Communication and Automation Technologies.*

Idasiak, V. & David, P., 2008. Towards a better interaction between design and dependability analysis: FMEA derived from UML/SysML models. *uropean Safety and Reliability Conference (ESREL).*

IEC 61508, 2010. *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems.*

Innal, F., Dutuit, Y., Rauzy, A. & Signoret, J.-P., July 2010. New insight into the average probability of failure on demand and the probability of dangerous failure per hour of safety instrumented systems. *Journal of Risk and Reliability,* Issue 224, pp. 75-86.

ISO 26262, 2011. *Road Vehicles - Functional Safety.* International Organization for Standardization / Technical Committee 22 (ISO/TC 22).

ITEA2, 2013. *Safe Project.* s.l.:http://www.safe-project.eu/.

Jin, H., Lundteigen, M.-A. & Rausand, M., 2011. Reliability performance of safety instrumented systems: A common approach for both low- and high-demand mode of operation. *Reliability Engineering and System Safety,* Issue 96, p. 365–373.

Leeman, M., 2013. The deployment of ISO26262 in Valeo. *SIA - Journée d'Etude Sécurité Fonctionnelle Electronique Automobile.*

L'Hostis, S., 2013. Architectural metrics calculation, an efficient approach. *SIA - Journée d'Etude Sécurité Fonctionnelle Electronique Automobile.*

Montgomery, A. & Marko, K., 1997. Quantitative FMEA Automation. *Reliability and Maintainability Symposium,* pp. 226-228.

Montgomery, A. & Pugh, D., 1996. FMEA Automation for the Complete design Process. *Reliability and Maintainability Symposium,* pp. 30-36.

Myers, A. & Rauzy, A., 2008. Assessment of Redundant Systems with Imperfect Coverage by means of Binary Decision Diagrams. *Reliability Engineering and System Safety,* 93(7), p. 1025–1035.

Page, R. & Rose, L., 2009. FPTC: Automated Safety Analysis for Domain-Specific Languages. *Lecture Notes in Computer Science (LNCS),* Volume 5421, pp. 229-242.

Papadopoulos, Y. & Parker, D., 2004. A Method and tool for Model Based Semi-automated FMEA of Engineering Designs. *Safety Critical Systems and Software.*

Papadopoulos, Y. & Parker, D., 2005. Automating FMEA of Safety critical Systems. *High Assurance Systems Engineering.*

Perrot, B., Prosvirnova, T. & Rauzy, A., 2010. *Dynamic Fault Trees: a library for AltaRica Next Generation.* s.l., LM17.

Price, C., Pugh, D., Wilson, N. & Snook, N., 1995. The Flame System : Automated Electrical FMEA. *Reliability and Maintainability Symposium,* pp. 90-95.

Prosvirnova, T., Batteux, M. & Brameret, P.-A., 2013. *The AltaRica 3.0 project for Model-Based Safety Assessment.* York, DCDS.

Rauzy, A., 2004. An Experimental Study on Six Algorithms to Compute Transient Solutions of Large Markov Systems. *Reliability Engineering and System Safety,* Issue 86, pp. 105-115.

Rauzy, A., 2012. *XFTA - An OpenPSA Fault Tree Engine.* s.l.:s.n.

Stanyer, T., 2012. *A practical Approach to Calculation of ISO 26262 Metrics for ASIL C/D Critical Systems.* s.l.: CTI Conference.

Teoh, P. & Case, K., 2004. Failure mode and effects analysis through knowledge modeling. *Journal of Materials Processing Technology,* Volume 153-154, pp. 253-260.

Torrente, G. & Bouissou, M., 2008. *Building Knowledge Bases in the Dependability Field with Open-source Environment "Visual Figaro".* Avignon, 16ème colloque de fiabilité et maintenabilité.

Wang, D. & Pan, J., 2010. An automatic FMEA technique for processes defined in the the Little-JIL Process definition. *International Conference on Software Engineering & Knowledge Engineering.*

Winkler, W. E., 2006. *Overview of Record Linkage and Current Research Directions,* Washington, DC: U.S. Census Bureau.

Winkovich, T. & Eckardt, D., 2005. Reliability analysis of safety systems using Markov-chain modelling. *European Conference on Power Electronics and Applications,* pp. 10-20.

Yoshimura, I., Sato, Y. & Suyama, K., 2004. Safety-Integrity Level model for safety-related systems in dynamic demand state. *Proceedings of Asian International Workshop on Advanced Reliability Modeling,* p. 577 – 584.

Zhang, T., Long, W. & Sato, Y., 2003. Availability of systems with self-diagnostic components -- applying Markov model to IEC 61508-6. *Reliability Engineering and System Safety,* Issue 80, pp. 133-141.